libUPnP

1.14.20

Generated by Doxygen 1.9.8

1 API changes after 1.10 1
1.1 Version 1.12 . . . 1
1.1.1 list.h header changed . . . 1
1.1.2 Removal of TRUE, FALSE macros and BOOL type . . . 1
1.2 Version 1.14 . . . 2
1.2.1 UpnpInit() has been dropped . . . 2
1.2.2 Some include files have been renamed . . . 2

2 Deprecated List 3

3 Todo List 5

4 Topic Index 7
4.1 Topics . . . 7

5 Data Structure Index 9
5.1 Data Structures . . . 9

6 File Index 13
6.1 File List . . . 13

7 Topic Documentation 17
7.1 UPnP API . . . 17
7.1.1 Detailed Description . . . 26
7.1.2 Macro Definition Documentation . . . 26
7.1.2.1 IN6_IS_ADDR_GLOBAL . . . 26
7.1.2.2 IN6_IS_ADDR_ULA . . . 26
7.1.2.3 UPNP_E_ALREADY_REGISTERED . . . 27
7.1.2.4 UPNP_E_BAD_HTTPMSG . . . 27
7.1.2.5 UPNP_E_BAD_RESPONSE . . . 27
7.1.2.6 UPNP_E_CANCELED . . . 27
7.1.2.7 UPNP_E_EXT_NOT_XML . . . 27
7.1.2.8 UPNP_E_FILE_NOT_FOUND . . . 27
7.1.2.9 UPNP_E_FILE_READ_ERROR . . . 28
7.1.2.10 UPNP_E_FINISH . . . 28
7.1.2.11 UPNP_E_INIT . . . 28
7.1.2.12 UPNP_E_INIT_FAILED . . . 28
7.1.2.13 UPNP_E_INTERNAL_ERROR . . . 28
7.1.2.14 UPNP_E_INVALID_ACTION . . . 28
7.1.2.15 UPNP_E_INVALID_ARGUMENT . . . 29
7.1.2.16 UPNP_E_INVALID_DESC . . . 29
7.1.2.17 UPNP_E_INVALID_HANDLE . . . 29
7.1.2.18 UPNP_E_INVALID_INTERFACE . . . 29
7.1.2.19 UPNP_E_INVALID_PARAM . . . 29

7.1.2.20 UPNP_E_INVALID_SERVICE . . . 29
7.1.2.21 UPNP_E_INVALID_URL . . . 30
7.1.2.22 UPNP_E_LISTEN . . . 30
7.1.2.23 UPNP_E_NETWORK_ERROR . . . 30
7.1.2.24 UPNP_E_NOT_FOUND . . . 30
7.1.2.25 UPNP_E_NOTIFY_UNACCEPTED . . . 30
7.1.2.26 UPNP_E_OUTOF_HANDLE . . . 30
7.1.2.27 UPNP_E_OUTOF_MEMORY . . . 31
7.1.2.28 UPNP_E_OUTOF_SOCKET . . . 31
7.1.2.29 UPNP_E_SOCKET_BIND . . . 31
7.1.2.30 UPNP_E_SOCKET_CONNECT . . . 31
7.1.2.31 UPNP_E_SOCKET_ERROR . . . 31
7.1.2.32 UPNP_E_SOCKET_READ . . . 32
7.1.2.33 UPNP_E_SOCKET_WRITE . . . 32
7.1.2.34 UPNP_E_SUBSCRIBE_UNACCEPTED . . . 32
7.1.2.35 UPNP_E_SUCCESS . . . 32
7.1.2.36 UPNP_E_TIMEDOUT . . . 32
7.1.2.37 UPNP_E_UNSUBSCRIBE_UNACCEPTED . . . 32
7.1.2.38 UPNP_E_URL_TOO_BIG . . . 33
7.1.3 Typedef Documentation . . . 33
7.1.3.1 Upnp_SID . . . 33
7.1.3.2 UpnpClient_Handle . . . 33
7.1.3.3 UpnpDevice_Handle . . . 33
7.1.3.4 UpnpWebFileHandle . . . 33
7.1.3.5 VDCallback_Close . . . 33
7.1.3.6 VDCallback_GetInfo . . . 34
7.1.3.7 VDCallback_Open . . . 34
7.1.3.8 VDCallback_Read . . . 34
7.1.3.9 VDCallback_Seek . . . 35
7.1.3.10 VDCallback_Write . . . 35
7.1.4 Enumeration Type Documentation . . . 35
7.1.4.1 Upnp_DescType_e . . . 35
7.1.4.2 Upnp_HttpMethod_e . . . 36
7.1.4.3 Upnp_SType_e . . . 36
7.1.5 Function Documentation . . . 36
7.1.5.1 AutoAdvertise() . . . 36
7.1.5.2 free_action_arg() . . . 36
7.1.5.3 free_advertise_arg() . . . 37
7.1.5.4 FreeHandle() . . . 37
7.1.5.5 GetCallBackFn() . . . 37
7.1.5.6 GetClientHandleInfo() . . . 37
7.1.5.7 GetDescDocumentAndURL() . . . 38

7.1.5.8 GetDeviceHandleInfo() . . . 38
7.1.5.9 GetDeviceHandleInfoForPath() . . . 39
7.1.5.10 GetFreeHandle() . . . 39
7.1.5.11 GetHandleInfo() . . . 39
7.1.5.12 PrintHandleInfo() . . . 40
7.1.5.13 PrintThreadPoolStats() . . . 40
7.1.5.14 UpnpAcceptSubscription() . . . 41
7.1.5.15 UpnpAcceptSubscriptionExt() . . . 42
7.1.5.16 UpnpAddVirtualDir() . . . 42
7.1.5.17 UpnpCancelHttpGet() . . . 43
7.1.5.18 UpnpCloseHttpConnection() . . . 43
7.1.5.19 UpnpCloseHttpGet() . . . 44
7.1.5.20 UpnpCloseHttpPost() . . . 44
7.1.5.21 UpnpComputeIpv6PrefixLength() . . . 45
7.1.5.22 UpnpDownloadUrlItem() . . . 45
7.1.5.23 UpnpDownloadXmlDoc() . . . 46
7.1.5.24 UpnpEnableWebserver() . . . 47
7.1.5.25 UpnpEndHttpRequest() . . . 47
7.1.5.26 UpnpFinish() . . . 48
7.1.5.27 UpnpGetHttpResponse() . . . 49
7.1.5.28 UpnpGetIfInfo() . . . 49
7.1.5.29 UpnpGetServerIp6Address() . . . 50
7.1.5.30 UpnpGetServerIpAddress() . . . 51
7.1.5.31 UpnpGetServerPort() . . . 51
7.1.5.32 UpnpGetServerPort6() . . . 52
7.1.5.33 UpnpGetServerUlaGuaIp6Address() . . . 52
7.1.5.34 UpnpGetServerUlaGuaPort6() . . . 52
7.1.5.35 UpnpGetServiceVarStatus() . . . 53
7.1.5.36 UpnpGetServiceVarStatusAsync() . . . 53
7.1.5.37 UpnpHttpGetProgress() . . . 54
7.1.5.38 UpnpInit2() . . . 55
7.1.5.39 UpnpInitMutexes() . . . 55
7.1.5.40 UpnpInitPreamble() . . . 56
7.1.5.41 UpnpInitStartServers() . . . 56
7.1.5.42 UpnpInitThreadPools() . . . 57
7.1.5.43 UpnpIsWebserverEnabled() . . . 57
7.1.5.44 UpnpMakeHttpRequest() . . . 58
7.1.5.45 UpnpNotify() . . . 59
7.1.5.46 UpnpNotifyExt() . . . 59
7.1.5.47 UpnpOpenHttpConnection() . . . 60
7.1.5.48 UpnpOpenHttpGet() . . . 61
7.1.5.49 UpnpOpenHttpGetEx() . . . 62

7.1.5.50 UpnpOpenHttpGetProxy() . . . 63
7.1.5.51 UpnpOpenHttpPost() . . . 64
7.1.5.52 UpnpReadHttpGet() . . . 64
7.1.5.53 UpnpReadHttpResponse() . . . 65
7.1.5.54 UpnpRegisterClient() . . . 66
7.1.5.55 UpnpRegisterRootDevice() . . . 66
7.1.5.56 UpnpRegisterRootDevice2() . . . 67
7.1.5.57 UpnpRegisterRootDevice3() . . . 69
7.1.5.58 UpnpRegisterRootDevice4() . . . 70
7.1.5.59 UpnpRemoveAllVirtualDirs() . . . 71
7.1.5.60 UpnpRemoveVirtualDir() . . . 71
7.1.5.61 UpnpRenewSubscription() . . . 71
7.1.5.62 UpnpRenewSubscriptionAsync() . . . 72
7.1.5.63 UpnpSearchAsync() . . . 73
7.1.5.64 UpnpSendAction() . . . 74
7.1.5.65 UpnpSendActionAsync() . . . 75
7.1.5.66 UpnpSendActionEx() . . . 76
7.1.5.67 UpnpSendActionExAsync() . . . 77
7.1.5.68 UpnpSendAdvertisement() . . . 77
7.1.5.69 UpnpSendAdvertisementLowPower() . . . 78
7.1.5.70 UpnpSetContentLength() . . . 79
7.1.5.71 UpnpSetMaxContentLength() . . . 79
7.1.5.72 UpnpSetMaxSubscriptions() . . . 80
7.1.5.73 UpnpSetMaxSubscriptionTimeOut() . . . 80
7.1.5.74 UpnpSetWebServerCorsString() . . . 81
7.1.5.75 UpnpSetWebServerRootDir() . . . 81
7.1.5.76 UpnpSubscribe() . . . 82
7.1.5.77 UpnpSubscribeAsync() . . . 83
7.1.5.78 UpnpThreadDistribution() . . . 84
7.1.5.79 UpnpUnRegisterClient() . . . 84
7.1.5.80 UpnpUnRegisterRootDevice() . . . 85
7.1.5.81 UpnpUnRegisterRootDeviceLowPower() . . . 85
7.1.5.82 UpnpUnSubscribe() . . . 86
7.1.5.83 UpnpUnSubscribeAsync() . . . 87
7.1.5.84 UpnpVirtualDir_set_CloseCallback() . . . 88
7.1.5.85 UpnpVirtualDir_set_GetInfoCallback() . . . 88
7.1.5.86 UpnpVirtualDir_set_OpenCallback() . . . 88
7.1.5.87 UpnpVirtualDir_set_ReadCallback() . . . 89
7.1.5.88 UpnpVirtualDir_set_SeekCallback() . . . 89
7.1.5.89 UpnpVirtualDir_set_WriteCallback() . . . 89
7.1.5.90 UpnpWriteHttpPost() . . . 90
7.1.5.91 UpnpWriteHttpRequest() . . . 90

7.1.5.92 WinsockInit() . . . 91
7.1.6 Variable Documentation . . . 91
7.1.6.1 bWebServerState . . . 91
7.1.6.2 g_maxContentLength . . . 91
7.1.6.3 g_UpnpSdkEQMaxAge . . . 92
7.1.6.4 g_UpnpSdkEQMaxLen . . . 92
7.1.6.5 gAllowLiteralHostRedirection . . . 92
7.1.6.6 gDocumentRootDir . . . 92
7.1.6.7 gIF_INDEX . . . 92
7.1.6.8 gIF_IPV4 . . . 92
7.1.6.9 gIF_IPV4_NETMASK . . . 93
7.1.6.10 gIF_IPV6 . . . 93
7.1.6.11 gIF_IPV6_PREFIX_LENGTH . . . 93
7.1.6.12 gIF_IPV6_ULA_GUA . . . 93
7.1.6.13 gIF_IPV6_ULA_GUA_PREFIX_LENGTH . . . 93
7.1.6.14 gIF_NAME . . . 93
7.1.6.15 GlobalClientSubscribeMutex . . . 94
7.1.6.16 GlobalHndRWLock . . . 94
7.1.6.17 gMiniServerThreadPool . . . 94
7.1.6.18 gRecvThreadPool . . . 94
7.1.6.19 gSDKInitMutex . . . 94
7.1.6.20 gSendThreadPool . . . 94
7.1.6.21 gTimerThread . . . 95
7.1.6.22 gUUIDMutex . . . 95
7.1.6.23 gWebCallback_HostValidate . . . 95
7.1.6.24 gWebCallback_HostValidateCookie . . . 95
7.1.6.25 gWebserverCorsString . . . 95
7.1.6.26 HandleTable . . . 95
7.1.6.27 LOCAL_PORT_V4 . . . 96
7.1.6.28 LOCAL_PORT_V6 . . . 96
7.1.6.29 LOCAL_PORT_V6_ULA_GUA . . . 96
7.1.6.30 pVirtualDirList . . . 96
7.1.6.31 UpnpSdkClientRegistered . . . 96
7.1.6.32 UpnpSdkDeviceRegisteredV4 . . . 96
7.1.6.33 UpnpSdkDeviceregisteredV6 . . . 97
7.1.6.34 UpnpSdkInit . . . 97
7.1.6.35 virtualDirCallback . . . 97
7.2 The UpnpString Class . . . 97
7.2.1 Detailed Description . . . 99
7.2.2 Typedef Documentation . . . 99
7.2.2.1 UpnpString . . . 99
7.2.3 Function Documentation . . . 99

7.2.3.1 UpnpString_assign() . . . 99
7.2.3.2 UpnpString_casecmp() . . . 99
7.2.3.3 UpnpString_clear() . . . 100
7.2.3.4 UpnpString_cmp() . . . 100
7.2.3.5 UpnpString_delete() . . . 101
7.2.3.6 UpnpString_dup() . . . 101
7.2.3.7 UpnpString_get_Length() . . . 101
7.2.3.8 UpnpString_get_String() . . . 102
7.2.3.9 UpnpString_new() . . . 103
7.2.3.10 UpnpString_set_Length() . . . 103
7.2.3.11 UpnpString_set_String() . . . 103
7.2.3.12 UpnpString_set_StringN() . . . 104
7.2.4 Variable Documentation . . . 105
7.2.4.1 m_length . . . 105
7.2.4.2 m_string . . . 105
7.3 Optional Tool API . . . 105
7.3.1 Detailed Description . . . 106
7.3.2 Function Documentation . . . 106
7.3.2.1 UpnpAddToAction() . . . 106
7.3.2.2 UpnpAddToActionResponse() . . . 107
7.3.2.3 UpnpAddToPropertySet() . . . 107
7.3.2.4 UpnpCreatePropertySet() . . . 108
7.3.2.5 UpnpGetErrorMessage() . . . 108
7.3.2.6 UpnpMakeAction() . . . 109
7.3.2.7 UpnpMakeActionResponse() . . . 109
7.3.2.8 UpnpResolveURL() . . . 110
7.3.2.9 UpnpResolveURL2() . . . 111
7.3.2.10 UpnpSetMaxJobsTotal() . . . 111
7.4 Sample Code . . . 112
7.4.1 Detailed Description . . . 118
7.4.2 Macro Definition Documentation . . . 118
7.4.2.1 MAX_BRIGHTNESS [1/2] . . . 118
7.4.2.2 MAX_BRIGHTNESS [2/2] . . . 118
7.4.2.3 MAX_CHANNEL [1/2] . . . 118
7.4.2.4 MAX_CHANNEL [2/2] . . . 119
7.4.2.5 MAX_COLOR [1/2] . . . 119
7.4.2.6 MAX_COLOR [2/2] . . . 119
7.4.2.7 MAX_CONTRAST [1/2] . . . 119
7.4.2.8 MAX_CONTRAST [2/2] . . . 119
7.4.2.9 MAX_TINT [1/2] . . . 119
7.4.2.10 MAX_TINT [2/2] . . . 119
7.4.2.11 MAX_VOLUME [1/2] . . . 119

7.4.2.12 MAX_VOLUME [2/2] . . . 120
7.4.2.13 POWER_ON [1/2] . . . 120
7.4.2.14 POWER_ON [2/2] . . . 120
7.4.2.15 TV_CONTROL_CHANNEL [1/2] . . . 120
7.4.2.16 TV_CONTROL_CHANNEL [2/2] . . . 120
7.4.2.17 TV_CONTROL_POWER [1/2] . . . 120
7.4.2.18 TV_CONTROL_POWER [2/2] . . . 120
7.4.2.19 TV_CONTROL_VARCOUNT [1/2] . . . 120
7.4.2.20 TV_CONTROL_VARCOUNT [2/2] . . . 121
7.4.2.21 TV_CONTROL_VOLUME [1/2] . . . 121
7.4.2.22 TV_CONTROL_VOLUME [2/2] . . . 121
7.4.2.23 TV_MAX_VAL_LEN [1/2] . . . 121
7.4.2.24 TV_MAX_VAL_LEN [2/2] . . . 121
7.4.2.25 TV_MAXACTIONS . . . 121
7.4.2.26 TV_MAXVARS [1/2] . . . 121
7.4.2.27 TV_MAXVARS [2/2] . . . 121
7.4.2.28 TV_PICTURE_BRIGHTNESS [1/2] . . . 122
7.4.2.29 TV_PICTURE_BRIGHTNESS [2/2] . . . 122
7.4.2.30 TV_PICTURE_COLOR [1/2] . . . 122
7.4.2.31 TV_PICTURE_COLOR [2/2] . . . 122
7.4.2.32 TV_PICTURE_CONTRAST [1/2] . . . 122
7.4.2.33 TV_PICTURE_CONTRAST [2/2] . . . 122
7.4.2.34 TV_PICTURE_TINT [1/2] . . . 122
7.4.2.35 TV_PICTURE_TINT [2/2] . . . 122
7.4.2.36 TV_PICTURE_VARCOUNT [1/2] . . . 123
7.4.2.37 TV_PICTURE_VARCOUNT [2/2] . . . 123
7.4.2.38 TV_SERVICE_CONTROL [1/2] . . . 123
7.4.2.39 TV_SERVICE_CONTROL [2/2] . . . 123
7.4.2.40 TV_SERVICE_PICTURE [1/2] . . . 123
7.4.2.41 TV_SERVICE_PICTURE [2/2] . . . 123
7.4.2.42 TV_SERVICE_SERVCOUNT [1/2] . . . 123
7.4.2.43 TV_SERVICE_SERVCOUNT [2/2] . . . 123
7.4.3 Typedef Documentation . . . 123
7.4.3.1 print_string . . . 123
7.4.3.2 state_update . . . 124
7.4.3.3 upnp_action . . . 124
7.4.4 Enumeration Type Documentation . . . 124
7.4.4.1 cmdloop_tvcmds . . . 124
7.4.5 Function Documentation . . . 125
7.4.5.1 device_main() . . . 125
7.4.5.2 IncrementBrightness() . . . 125
7.4.5.3 IncrementColor() . . . 125

7.4.5.4 IncrementContrast() . . . 126
7.4.5.5 IncrementVolume() . . . 126
7.4.5.6 linux_print() . . . 127
7.4.5.7 SampleUtil_FindAndParseService() . . . 127
7.4.5.8 SampleUtil_Finish() . . . 128
7.4.5.9 SampleUtil_GetElementValue() . . . 128
7.4.5.10 SampleUtil_GetFirstDocumentItem() . . . 128
7.4.5.11 SampleUtil_GetFirstElementItem() . . . 129
7.4.5.12 SampleUtil_GetFirstServiceList() . . . 129
7.4.5.13 SampleUtil_Initialize() . . . 129
7.4.5.14 SampleUtil_Print() . . . 130
7.4.5.15 SampleUtil_PrintEvent() . . . 130
7.4.5.16 SampleUtil_PrintEventType() . . . 131
7.4.5.17 SampleUtil_RegisterUpdateFunction() . . . 131
7.4.5.18 SampleUtil_StateUpdate() . . . 132
7.4.5.19 SetActionTable() . . . 132
7.4.5.20 SetServiceTable() . . . 132
7.4.5.21 TvCtrlPointCommandLoop() . . . 134
7.4.5.22 TvCtrlPointPrintCommands() . . . 134
7.4.5.23 TvCtrlPointPrintLongHelp() . . . 134
7.4.5.24 TvCtrlPointPrintShortHelp() . . . 134
7.4.5.25 TvCtrlPointStart() . . . 135
7.4.5.26 TvCtrlPointVerifyTimeouts() . . . 135
7.4.5.27 TvDeviceCallbackEventHandler() . . . 135
7.4.5.28 TvDeviceCommandLoop() . . . 136
7.4.5.29 TvDeviceDecreaseBrightness() . . . 136
7.4.5.30 TvDeviceDecreaseChannel() . . . 136
7.4.5.31 TvDeviceDecreaseColor() . . . 138
7.4.5.32 TvDeviceDecreaseContrast() . . . 138
7.4.5.33 TvDeviceDecreaseTint() . . . 138
7.4.5.34 TvDeviceDecreaseVolume() . . . 139
7.4.5.35 TvDeviceHandleActionRequest() . . . 139
7.4.5.36 TvDeviceHandleGetVarRequest() . . . 140
7.4.5.37 TvDeviceHandleSubscriptionRequest() . . . 140
7.4.5.38 TvDeviceIncreaseBrightness() . . . 140
7.4.5.39 TvDeviceIncreaseChannel() . . . 141
7.4.5.40 TvDeviceIncreaseColor() . . . 141
7.4.5.41 TvDeviceIncreaseContrast() . . . 142
7.4.5.42 TvDeviceIncreaseTint() . . . 142
7.4.5.43 TvDeviceIncreaseVolume() . . . 142
7.4.5.44 TvDevicePowerOff() . . . 143
7.4.5.45 TvDevicePowerOn() . . . 143

7.4.5.46 TvDeviceSetBrightness() . . . 143
7.4.5.47 TvDeviceSetChannel() . . . 144
7.4.5.48 TvDeviceSetColor() . . . 144
7.4.5.49 TvDeviceSetContrast() . . . 145
7.4.5.50 TvDeviceSetPower() . . . 145
7.4.5.51 TvDeviceSetServiceTableVar() . . . 146
7.4.5.52 TvDeviceSetTint() . . . 146
7.4.5.53 TvDeviceSetVolume() . . . 147
7.4.5.54 TvDeviceStart() . . . 147
7.4.5.55 TvDeviceStateTableInit() . . . 148
7.4.5.56 TvDeviceStop() . . . 148
7.4.5.57 TvStateUpdate() . . . 148
7.4.6 Variable Documentation . . . 149
7.4.6.1 cmdloop_cmdlist . . . 149
7.4.6.2 default_advr_expire . . . 149
7.4.6.3 default_timeout . . . 149
7.4.6.4 device_handle [1/2] . . . 149
7.4.6.5 device_handle [2/2] . . . 150
7.4.6.6 DeviceListMutex [1/2] . . . 150
7.4.6.7 DeviceListMutex [2/2] . . . 150
7.4.6.8 display_mutex [1/2] . . . 150
7.4.6.9 display_mutex [2/2] . . . 150
7.4.6.10 GlobalDeviceList [1/2] . . . 151
7.4.6.11 GlobalDeviceList [2/2] . . . 151
7.4.6.12 gPrintFun [1/2] . . . 151
7.4.6.13 gPrintFun [2/2] . . . 151
7.4.6.14 gStateUpdateFun [1/2] . . . 151
7.4.6.15 gStateUpdateFun [2/2] . . . 151
7.4.6.16 tv_service_table [1/2] . . . 152
7.4.6.17 tv_service_table [2/2] . . . 152
7.4.6.18 tvc_varname . . . 152
7.4.6.19 TvCtrlPointTimerLoopRun . . . 152
7.4.6.20 TvDeviceType . . . 152
7.4.6.21 TVDevMutex [1/2] . . . 152
7.4.6.22 TVDevMutex [2/2] . . . 153
7.4.6.23 tvp_varname . . . 153
7.4.6.24 TvServiceName [1/2] . . . 153
7.4.6.25 TvServiceName [2/2] . . . 153
7.4.6.26 TvVarCount . . . 153
7.4.6.27 TvVarName [1/2] . . . 153
7.4.6.28 TvVarName [2/2] . . . 154
7.5 Network Socket Library . . . 154

7.5.1 Detailed Description . . . 155
7.5.2 Macro Definition Documentation . . . 155
7.5.2.1 INVALID_SOCKET . . . 155
7.5.2.2 SOCKET_ERROR . . . 155
7.5.2.3 UpnpCloseSocket . . . 155
7.5.3 Typedef Documentation . . . 155
7.5.3.1 SOCKET . . . 155
7.5.4 Function Documentation . . . 155
7.5.4.1 sock_close() . . . 155
7.5.4.2 sock_destroy() . . . 156
7.5.4.3 sock_init() . . . 156
7.5.4.4 sock_init_with_ip() . . . 157
7.5.4.5 sock_make_blocking() . . . 157
7.5.4.6 sock_make_no_blocking() . . . 158
7.5.4.7 sock_read() . . . 158
7.5.4.8 sock_read_write() . . . 158
7.5.4.9 sock_write() . . . 159
7.5.5 Variable Documentation . . . 160
7.5.5.1 foreign_sockaddr . . . 160
7.5.5.2 socket . . . 160
7.6 SSDP Library . . . 160
7.6.1 Detailed Description . . . 164
7.6.2 Macro Definition Documentation . . . 164
7.6.2.1 NO_ERROR_FOUND . . . 164
7.6.3 Typedef Documentation . . . 164
7.6.3.1 SsdpEvent . . . 164
7.6.3.2 SType . . . 164
7.6.4 Enumeration Type Documentation . . . 164
7.6.4.1 SsdpSearchType . . . 164
7.6.5 Function Documentation . . . 165
7.6.5.1 AdvertiseAndReply() . . . 165
7.6.5.2 advertiseAndReplyThread() . . . 165
7.6.5.3 create_ssdp_sock_reqv4() . . . 165
7.6.5.4 create_ssdp_sock_v4() . . . 166
7.6.5.5 CreateClientRequestPacket() . . . 166
7.6.5.6 CreateServicePacket() . . . 166
7.6.5.7 DeviceAdvertisement() . . . 167
7.6.5.8 DeviceReply() . . . 168
7.6.5.9 DeviceShutdown() . . . 168
7.6.5.10 extractIPv6address() . . . 168
7.6.5.11 free_ssdp_event_handler_data() . . . 169
7.6.5.12 get_ssdp_sockets() . . . 169

7.6.5.13 isUrlV6UlaGua() . . . 169
7.6.5.14 NewRequestHandler() . . . 169
7.6.5.15 readFromSSDPSocket() . . . 170
7.6.5.16 SearchByTarget() . . . 170
7.6.5.17 send_search_result() . . . 171
7.6.5.18 SendReply() . . . 171
7.6.5.19 ServiceAdvertisement() . . . 171
7.6.5.20 ServiceReply() . . . 172
7.6.5.21 ServiceShutdown() . . . 172
7.6.5.22 ssdp_event_handler_thread() . . . 172
7.6.5.23 ssdp_handle_ctrlpt_msg() . . . 173
7.6.5.24 ssdp_handle_device_request() . . . 173
7.6.5.25 ssdp_request_type() . . . 174
7.6.5.26 ssdp_request_type1() . . . 174
7.6.5.27 start_event_handler() . . . 174
7.6.5.28 unique_service_name() . . . 175
7.6.5.29 valid_ssdp_msg() . . . 175
7.6.6 Variable Documentation . . . 175
7.6.6.1 ssdpReqSock . . . 175
7.6.6.2 ssdpSock . . . 175
7.7 XML API . . . 176
7.7.1 Detailed Description . . . 183
7.7.2 Macro Definition Documentation . . . 183
7.7.2.1 DOMString . . . 183
7.7.3 Typedef Documentation . . . 183
7.7.3.1 IXML_Attr . . . 183
7.7.3.2 IXML_CDATASection . . . 184
7.7.3.3 IXML_Document . . . 184
7.7.3.4 IXML_Element . . . 184
7.7.3.5 IXML_NamedNodeMap . . . 184
7.7.3.6 IXML_Node . . . 184
7.7.3.7 IXML_NodeList . . . 184
7.7.3.8 IXML_Text . . . 184
7.7.4 Enumeration Type Documentation . . . 185
7.7.4.1 IXML_ERRORCODE . . . 185
7.7.4.2 IXML_NODE_TYPE . . . 185
7.7.5 Function Documentation . . . 185
7.7.5.1 ixmlAttr_free() . . . 185
7.7.5.2 ixmlCDATASection_free() . . . 185
7.7.5.3 ixmlCDATASection_init() . . . 186
7.7.5.4 ixmlCloneDOMString() . . . 186
7.7.5.5 ixmlDocument_createAttribute() . . . 186

7.7.5.6 ixmlDocument_createAttributeEx() . . . 187
7.7.5.7 ixmlDocument_createAttributeNS() . . . 187
7.7.5.8 ixmlDocument_createAttributeNSEx() . . . 188
7.7.5.9 ixmlDocument_createCDATASection() . . . 188
7.7.5.10 ixmlDocument_createCDATASectionEx() . . . 189
7.7.5.11 ixmlDocument_createDocument() . . . 189
7.7.5.12 ixmlDocument_createDocumentEx() . . . 189
7.7.5.13 ixmlDocument_createElement() . . . 190
7.7.5.14 ixmlDocument_createElementEx() . . . 190
7.7.5.15 ixmlDocument_createElementNS() . . . 191
7.7.5.16 ixmlDocument_createElementNSEx() . . . 191
7.7.5.17 ixmlDocument_createTextNode() . . . 192
7.7.5.18 ixmlDocument_createTextNodeEx() . . . 192
7.7.5.19 ixmlDocument_free() . . . 193
7.7.5.20 ixmlDocument_getElementById() . . . 193
7.7.5.21 ixmlDocument_getElementsByTagName() . . . 193
7.7.5.22 ixmlDocument_getElementsByTagNameNS() . . . 194
7.7.5.23 ixmlDocument_importNode() . . . 194
7.7.5.24 ixmlDocument_init() . . . 195
7.7.5.25 ixmlDocumenttoString() . . . 195
7.7.5.26 ixmlElement_free() . . . 196
7.7.5.27 ixmlElement_getAttribute() . . . 196
7.7.5.28 ixmlElement_getAttributeNode() . . . 196
7.7.5.29 ixmlElement_getAttributeNodeNS() . . . 197
7.7.5.30 ixmlElement_getAttributeNS() . . . 197
7.7.5.31 ixmlElement_getElementsByTagName() . . . 198
7.7.5.32 ixmlElement_getElementsByTagNameNS() . . . 198
7.7.5.33 ixmlElement_getTagName() . . . 198
7.7.5.34 ixmlElement_hasAttribute() . . . 199
7.7.5.35 ixmlElement_hasAttributeNS() . . . 199
7.7.5.36 ixmlElement_init() . . . 200
7.7.5.37 ixmlElement_removeAttribute() . . . 200
7.7.5.38 ixmlElement_removeAttributeNode() . . . 200
7.7.5.39 ixmlElement_removeAttributeNS() . . . 201
7.7.5.40 ixmlElement_setAttribute() . . . 201
7.7.5.41 ixmlElement_setAttributeNode() . . . 202
7.7.5.42 ixmlElement_setAttributeNodeNS() . . . 202
7.7.5.43 ixmlElement_setAttributeNS() . . . 203
7.7.5.44 ixmlFreeDOMString() . . . 204
7.7.5.45 ixmlLoadDocument() . . . 204
7.7.5.46 ixmlLoadDocumentEx() . . . 204
7.7.5.47 ixmlNamedNodeMap_free() . . . 205

7.7.5.48 ixmlNamedNodeMap_getLength() . . . 205
7.7.5.49 ixmlNamedNodeMap_getNamedItem() . . . 205
7.7.5.50 ixmlNamedNodeMap_getNamedItemNS() . . . 206
7.7.5.51 ixmlNamedNodeMap_item() . . . 206
7.7.5.52 ixmlNamedNodeMap_removeNamedItem() . . . 207
7.7.5.53 ixmlNamedNodeMap_removeNamedItemNS() . . . 207
7.7.5.54 ixmlNamedNodeMap_setNamedItem() . . . 207
7.7.5.55 ixmlNamedNodeMap_setNamedItemNS() . . . 208
7.7.5.56 ixmlNode_appendChild() . . . 208
7.7.5.57 ixmlNode_cloneNode() . . . 209
7.7.5.58 ixmlNode_free() . . . 209
7.7.5.59 ixmlNode_getAttributes() . . . 210
7.7.5.60 ixmlNode_getChildNodes() . . . 210
7.7.5.61 ixmlNode_getFirstChild() . . . 210
7.7.5.62 ixmlNode_getLastChild() . . . 211
7.7.5.63 ixmlNode_getLocalName() . . . 211
7.7.5.64 ixmlNode_getNamespaceURI() . . . 212
7.7.5.65 ixmlNode_getNextSibling() . . . 212
7.7.5.66 ixmlNode_getNodeName() . . . 212
7.7.5.67 ixmlNode_getNodeType() . . . 213
7.7.5.68 ixmlNode_getNodeValue() . . . 213
7.7.5.69 ixmlNode_getOwnerDocument() . . . 214
7.7.5.70 ixmlNode_getParentNode() . . . 214
7.7.5.71 ixmlNode_getPrefix() . . . 214
7.7.5.72 ixmlNode_getPreviousSibling() . . . 215
7.7.5.73 ixmlNode_hasAttributes() . . . 215
7.7.5.74 ixmlNode_hasChildNodes() . . . 215
7.7.5.75 ixmlNode_insertBefore() . . . 216
7.7.5.76 ixmlNode_removeChild() . . . 216
7.7.5.77 ixmlNode_replaceChild() . . . 217
7.7.5.78 ixmlNode_setNodeValue() . . . 218
7.7.5.79 ixmlNodeList_free() . . . 218
7.7.5.80 ixmlNodeList_item() . . . 219
7.7.5.81 ixmlNodeList_length() . . . 219
7.7.5.82 ixmlNodetoString() . . . 219
7.7.5.83 ixmlParseBuffer() . . . 220
7.7.5.84 ixmlParseBufferEx() . . . 220
7.7.5.85 ixmlPrintDocument() . . . 221
7.7.5.86 ixmlPrintNode() . . . 221
7.7.5.87 ixmlRelaxParser() . . . 222

8 Data Structure Documentation **223**

8.1 _IXML_ATTR Struct Reference . . . 223
8.1.1 Detailed Description . . . 224
8.2 _IXML_CDATASection Struct Reference . . . 224
8.2.1 Detailed Description . . . 224
8.3 _IXML_Document Struct Reference . . . 225
8.3.1 Detailed Description . . . 225
8.4 _IXML_Element Struct Reference . . . 225
8.4.1 Detailed Description . . . 226
8.5 _IXML_ElementStack Struct Reference . . . 226
8.6 _IXML_NamedNodeMap Struct Reference . . . 227
8.6.1 Detailed Description . . . 227
8.7 _IXML_NamespaceURI Struct Reference . . . 228
8.8 _IXML_Node Struct Reference . . . 228
8.8.1 Detailed Description . . . 229
8.9 _IXML_NodeList Struct Reference . . . 229
8.9.1 Detailed Description . . . 230
8.10 _IXML_Text Struct Reference . . . 230
8.10.1 Detailed Description . . . 230
8.11 _Parser Struct Reference . . . 231
8.11.1 Field Documentation . . . 231
8.11.1.1 curPtr . . . 231
8.11.1.2 dataBuffer . . . 231
8.11.1.3 savePtr . . . 232
8.12 _uuid_upnp Struct Reference . . . 232
8.12.1 Detailed Description . . . 232
8.12.2 Field Documentation . . . 232
8.12.2.1 clock_seq_hi_and_reserved . . . 232
8.12.2.2 clock_seq_low . . . 232
8.12.2.3 node . . . 232
8.12.2.4 time_hi_and_version . . . 233
8.12.2.5 time_low . . . 233
8.12.2.6 time_mid . . . 233
8.13 char_info Struct Reference . . . 233
8.14 cmdloop_commands Struct Reference . . . 233
8.14.1 Detailed Description . . . 234
8.15 document_type_t Struct Reference . . . 234
8.15.1 Field Documentation . . . 234
8.15.1.1 content_subtype . . . 234
8.15.1.2 content_type . . . 234
8.15.1.3 file_ext . . . 234
8.16 ErrorString Struct Reference . . . 234
8.16.1 Detailed Description . . . 235

8.16.2 Field Documentation . . . 235
8.16.2.1 rc . . . 235
8.16.2.2 rcError . . . 235
8.17 FREELIST Struct Reference . . . 235
8.17.1 Detailed Description . . . 236
8.18 FREELISTNODE Struct Reference . . . 236
8.18.1 Detailed Description . . . 236
8.19 Handle_Info Struct Reference . . . 237
8.19.1 Field Documentation . . . 238
8.19.1.1 aliasInstalled . . . 238
8.19.1.2 Callback . . . 238
8.19.1.3 ClientSubList . . . 238
8.19.1.4 Cookie . . . 238
8.19.1.5 DescDocument . . . 238
8.19.1.6 DescURL . . . 239
8.19.1.7 DescXML . . . 239
8.19.1.8 DeviceAf . . . 239
8.19.1.9 DeviceList . . . 239
8.19.1.10 HType . . . 239
8.19.1.11 LowerDescURL . . . 239
8.19.1.12 MaxSubscriptions . . . 240
8.19.1.13 MaxSubscriptionTimeOut . . . 240
8.19.1.14 ServiceList . . . 240
8.19.1.15 ServiceTable . . . 240
8.19.1.16 SsdpSearchList . . . 240
8.20 HOSTPORT Struct Reference . . . 241
8.20.1 Detailed Description . . . 241
8.20.2 Field Documentation . . . 241
8.20.2.1 text . . . 241
8.21 http_header_t Struct Reference . . . 242
8.21.1 Field Documentation . . . 242
8.21.1.1 name . . . 242
8.21.1.2 name_id . . . 242
8.21.1.3 value . . . 242
8.22 http_message_t Struct Reference . . . 243
8.22.1 Field Documentation . . . 243
8.22.1.1 amount_discarded . . . 243
8.22.1.2 entity . . . 244
8.22.1.3 headers . . . 244
8.22.1.4 is_request . . . 244
8.22.1.5 method . . . 244
8.22.1.6 msg . . . 244

8.22.1.7 request_method . . . 244
8.22.1.8 status_code . . . 245
8.22.1.9 status_msg . . . 245
8.22.1.10 uri . . . 245
8.22.1.11 urlbuf . . . 245
8.23 http_parser_t Struct Reference . . . 246
8.23.1 Field Documentation . . . 246
8.23.1.1 entity_start_position . . . 246
8.23.1.2 http_error_code . . . 247
8.23.1.3 valid_ssdp_notify_hack . . . 247
8.24 HTTPCONNECTIONHANDLE Struct Reference . . . 247
8.25 ixml_membuf Struct Reference . . . 248
8.25.1 Detailed Description . . . 248
8.26 job_arg Struct Reference . . . 249
8.26.1 Detailed Description . . . 249
8.27 LINKEDLIST Struct Reference . . . 250
8.27.1 Detailed Description . . . 250
8.27.2 Field Documentation . . . 251
8.27.2.1 cmp_func . . . 251
8.27.2.2 free_func . . . 251
8.27.2.3 freeNodeList . . . 251
8.27.2.4 head . . . 251
8.27.2.5 size . . . 251
8.27.2.6 tail . . . 252
8.28 LISTNODE Struct Reference . . . 252
8.28.1 Detailed Description . . . 252
8.29 MD5Context Struct Reference . . . 253
8.30 membuffer Struct Reference . . . 253
8.30.1 Detailed Description . . . 253
8.30.2 Field Documentation . . . 253
8.30.2.1 buf . . . 253
8.30.2.2 capacity . . . 254
8.30.2.3 length . . . 254
8.30.2.4 size_inc . . . 254
8.31 memptr Struct Reference . . . 254
8.31.1 Detailed Description . . . 254
8.31.2 Field Documentation . . . 255
8.31.2.1 buf . . . 255
8.31.2.2 length . . . 255
8.32 mserv_request_t Struct Reference . . . 255
8.32.1 Field Documentation . . . 255
8.32.1.1 connfd . . . 255

8.32.1.2 foreign_sockaddr . . . 256
8.33 MServerSockArray Struct Reference . . . 256
8.33.1 Field Documentation . . . 256
8.33.1.1 miniServerSock4 . . . 256
8.33.1.2 miniServerSock6 . . . 256
8.33.1.3 miniServerSock6UlaGua . . . 256
8.33.1.4 miniServerStopSock . . . 257
8.33.1.5 ssdpReqSock4 . . . 257
8.33.1.6 ssdpReqSock6 . . . 257
8.33.1.7 ssdpSock4 . . . 257
8.33.1.8 ssdpSock6 . . . 257
8.33.1.9 ssdpSock6UlaGua . . . 257
8.34 NOTIFY_THREAD_STRUCT Struct Reference . . . 258
8.34.1 Detailed Description . . . 258
8.35 s_Class Struct Reference . . . 258
8.36 s_GenlibClientSubscription Struct Reference . . . 259
8.37 s_Member Struct Reference . . . 259
8.38 s_SSDPResultData Struct Reference . . . 260
8.39 s_TestClass Struct Reference . . . 260
8.40 s_Type_Integer Struct Reference . . . 261
8.41 s_UpnpActionComplete Struct Reference . . . 261
8.42 s_UpnpActionRequest Struct Reference . . . 262
8.43 s_UpnpDiscovery Struct Reference . . . 263
8.44 s_UpnpEvent Struct Reference . . . 263
8.45 s_UpnpEventSubscribe Struct Reference . . . 264
8.46 s_UpnpExtraHeaders Struct Reference . . . 264
8.47 s_UpnpFileInfo Struct Reference . . . 265
8.48 s_UpnpStateVarComplete Struct Reference . . . 265
8.49 s_UpnpStateVarRequest Struct Reference . . . 266
8.50 s_UpnpSubscriptionRequest Struct Reference . . . 266
8.51 scanner_t Struct Reference . . . 266
8.51.1 Field Documentation . . . 267
8.51.1.1 cursor . . . 267
8.51.1.2 entire_msg_loaded . . . 267
8.51.1.3 msg . . . 267
8.52 SendInstruction Struct Reference . . . 267
8.52.1 Field Documentation . . . 268
8.52.1.1 Cookie . . . 268
8.52.1.2 CorsHeader . . . 268
8.52.1.3 ReadSendSize . . . 268
8.52.1.4 RecvWriteSize . . . 268
8.52.1.5 RequestCookie . . . 268

8.53 SERVICE_INFO Struct Reference . . . 269
8.54 SERVICE_TABLE Struct Reference . . . 270
8.55 soap_devserv_t Struct Reference . . . 271
8.56 SOCKINFO Struct Reference . . . 271
8.57 ssdp_thread_data Struct Reference . . . 272
8.58 SsdpEventStruct Struct Reference . . . 272
8.58.1 Detailed Description . . . 273
8.59 ssdpsearcharg Struct Reference . . . 273
8.60 ssdpsearchexparg Struct Reference . . . 273
8.61 ssdpsearchreply Struct Reference . . . 274
8.62 SSDPSockArray Struct Reference . . . 274
8.63 str_int_entry Struct Reference . . . 274
8.64 SUBSCRIPTION Struct Reference . . . 275
8.65 SUpnpString Struct Reference . . . 275
8.65.1 Detailed Description . . . 276
8.66 TData Struct Reference . . . 276
8.67 TheStruct Struct Reference . . . 276
8.68 THREADPOOL Struct Reference . . . 277
8.68.1 Detailed Description . . . 277
8.68.2 Field Documentation . . . 278
8.68.2.1 attr . . . 278
8.68.2.2 busyThreads . . . 278
8.68.2.3 condition . . . 278
8.68.2.4 highJobQ . . . 278
8.68.2.5 jobFreeList . . . 278
8.68.2.6 lastJobId . . . 279
8.68.2.7 lowJobQ . . . 279
8.68.2.8 medJobQ . . . 279
8.68.2.9 mutex . . . 279
8.68.2.10 pendingWorkerThreadStart . . . 279
8.68.2.11 persistentJob . . . 279
8.68.2.12 persistentThreads . . . 280
8.68.2.13 shutdown . . . 280
8.68.2.14 start_and_shutdown . . . 280
8.68.2.15 stats . . . 280
8.68.2.16 totalThreads . . . 280
8.69 THREADPOOLATTR Struct Reference . . . 281
8.69.1 Detailed Description . . . 281
8.69.2 Field Documentation . . . 281
8.69.2.1 jobsPerThread . . . 281
8.69.2.2 maxIdleTime . . . 281
8.69.2.3 maxJobsTotal . . . 281

8.69.2.4 maxThreads . . . 282
8.69.2.5 minThreads . . . 282
8.69.2.6 schedPolicy . . . 282
8.69.2.7 stackSize . . . 282
8.69.2.8 starvationTime . . . 282
8.70 THREADPOOLJOB Struct Reference . . . 282
8.70.1 Detailed Description . . . 283
8.71 TIMEREVENT Struct Reference . . . 283
8.71.1 Detailed Description . . . 283
8.71.2 Field Documentation . . . 284
8.71.2.1 eventTime . . . 284
8.71.2.2 persistent . . . 284
8.72 TIMERTHREAD Struct Reference . . . 284
8.72.1 Detailed Description . . . 285
8.73 TOKEN Struct Reference . . . 285
8.73.1 Detailed Description . . . 285
8.74 TPOOLSTATS Struct Reference . . . 285
8.74.1 Detailed Description . . . 286
8.75 tv_service Struct Reference . . . 286
8.76 TvDevice Struct Reference . . . 287
8.77 TvDeviceNode Struct Reference . . . 287
8.78 TvService Struct Reference . . . 288
8.78.1 Detailed Description . . . 288
8.78.2 Field Documentation . . . 289
8.78.2.1 ActionNames . . . 289
8.78.2.2 actions . . . 289
8.78.2.3 ServiceId . . . 289
8.78.2.4 ServiceType . . . 289
8.78.2.5 UDN . . . 289
8.78.2.6 VariableCount . . . 289
8.78.2.7 VariableName . . . 289
8.78.2.8 VariableStrVal . . . 290
8.79 UpnpListHead Struct Reference . . . 290
8.79.1 Detailed Description . . . 290
8.80 UpnpNonblockParam Struct Reference . . . 291
8.81 URI Struct Reference . . . 292
8.81.1 Detailed Description . . . 292
8.82 URL_LIST Struct Reference . . . 293
8.82.1 Detailed Description . . . 293
8.82.2 Field Documentation . . . 294
8.82.2.1 URLs . . . 294
8.83 uuid_node_t Struct Reference . . . 294

8.84 uuid_state Struct Reference . . . 294
8.84.1 Detailed Description . . . 295
8.84.2 Field Documentation . . . 295
8.84.2.1 cs . . . 295
8.84.2.2 node . . . 295
8.84.2.3 ts . . . 295
8.85 virtual_Dir_List Struct Reference . . . 295
8.86 VirtualDirCallbacks Struct Reference . . . 296
8.86.1 Detailed Description . . . 296
8.86.2 Field Documentation . . . 297
8.86.2.1 close . . . 297
8.86.2.2 get_info . . . 297
8.86.2.3 open . . . 297
8.86.2.4 read . . . 297
8.86.2.5 seek . . . 298
8.86.2.6 write . . . 298
8.87 xml_alias_t Struct Reference . . . 298
8.87.1 Field Documentation . . . 299
8.87.1.1 ct . . . 299
8.87.1.2 doc . . . 299
8.87.1.3 last_modified . . . 299
8.87.1.4 name . . . 299

9 File Documentation **301**

9.1 /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h File Reference . . . 301
9.2 ixml.h . . . 308
9.3 /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixmldebug.h File Reference . . . 314
9.3.1 Detailed Description . . . 315
9.3.2 Function Documentation . . . 316
9.3.2.1 IxmlPrintf() . . . 316
9.3.2.2 printNodes() . . . 316
9.4 ixmldebug.h . . . 316
9.5 posix_overwrites.h . . . 317
9.6 /build/reproducible-path/pupnp-1.14.20/ixml/src/document.c File Reference . . . 318
9.6.1 Function Documentation . . . 319
9.6.1.1 ixmlDocument_setOwnerDocument() . . . 319
9.7 /build/reproducible-path/pupnp-1.14.20/ixml/src/element.c File Reference . . . 319
9.7.1 Function Documentation . . . 321
9.7.1.1 ixmlElement_findAttributeNode() . . . 321
9.7.1.2 ixmlElement_setTagName() . . . 321
9.8 /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlmembuf.h File Reference . . . 322
9.8.1 Function Documentation . . . 323

9.8.1.1 ixml_membuf_append() . . . 323
9.8.1.2 ixml_membuf_append_str() . . . 324
9.8.1.3 ixml_membuf_assign() . . . 324
9.8.1.4 ixml_membuf_assign_str() . . . 325
9.8.1.5 ixml_membuf_destroy() . . . 325
9.8.1.6 ixml_membuf_init() . . . 326
9.8.1.7 ixml_membuf_insert() . . . 326
9.9 ixmlmembuf.h . . . 326
9.10 /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlparser.h File Reference . . . 328
9.10.1 Function Documentation . . . 330
9.10.1.1 ixmlElement_setTagName() . . . 330
9.10.1.2 ixmlNamedNodeMap_addToNamedNodeMap() . . . 330
9.10.1.3 ixmlNamedNodeMap_init() . . . 330
9.10.1.4 ixmlNode_compare() . . . 331
9.10.1.5 ixmlNode_getElementsByTagName() . . . 331
9.10.1.6 ixmlNode_getElementsByTagNameNS() . . . 331
9.10.1.7 ixmlNode_init() . . . 332
9.10.1.8 ixmlNode_setNodeName() . . . 332
9.10.1.9 ixmlNode_setNodeProperties() . . . 333
9.10.1.10 ixmlNodeList_addToNodeList() . . . 333
9.10.1.11 ixmlNodeList_init() . . . 333
9.10.1.12 Parser_freeNodeContent() . . . 334
9.10.1.13 Parser_isValidXmlName() . . . 334
9.10.1.14 Parser_LoadDocument() . . . 334
9.10.1.15 Parser_setErrorChar() . . . 334
9.10.1.16 Parser_setNodePrefixAndLocalName() . . . 335
9.11 ixmlparser.h . . . 335
9.12 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixml.c File Reference . . . 337
9.12.1 Function Documentation . . . 338
9.12.1.1 copy_with_escape() . . . 338
9.12.1.2 ixmlDomTreetoString() . . . 339
9.12.1.3 ixmlPrintDomTree() . . . 339
9.12.1.4 ixmlPrintDomTreeRecursive() . . . 340
9.13 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixmldebug.c File Reference . . . 340
9.13.1 Function Documentation . . . 341
9.13.1.1 IxmlPrintf() . . . 341
9.13.1.2 printNodes() . . . 342
9.14 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlmembuf.c File Reference . . . 342
9.14.1 Function Documentation . . . 343
9.14.1.1 ixml_membuf_append() . . . 343
9.14.1.2 ixml_membuf_append_str() . . . 343
9.14.1.3 ixml_membuf_assign() . . . 344

9.14.1.4 ixml_membuf_assign_str() . . . 345
9.14.1.5 ixml_membuf_destroy() . . . 345
9.14.1.6 ixml_membuf_init() . . . 346
9.14.1.7 ixml_membuf_insert() . . . 346
9.14.1.8 ixml_membuf_set_size() . . . 346
9.15 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlparser.c File Reference . . . 347
9.15.1 Macro Definition Documentation . . . 351
9.15.1.1 LETTERTABLESIZE . . . 351
9.15.1.2 NAMECHARTABLESIZE . . . 351
9.15.2 Function Documentation . . . 351
9.15.2.1 isDuplicateAttribute() . . . 351
9.15.2.2 isTopLevelElement() . . . 351
9.15.2.3 Parser_addNamespace() . . . 352
9.15.2.4 Parser_appendTokBufChar() . . . 352
9.15.2.5 Parser_appendTokBufStr() . . . 352
9.15.2.6 Parser_clearTokenBuf() . . . 353
9.15.2.7 Parser_copyToken() . . . 353
9.15.2.8 Parser_ElementPrefixDefined() . . . 353
9.15.2.9 Parser_eTagVerification() . . . 354
9.15.2.10 Parser_free() . . . 354
9.15.2.11 Parser_freeElementStackItem() . . . 354
9.15.2.12 Parser_freeNodeContent() . . . 356
9.15.2.13 Parser_freeNsURI() . . . 356
9.15.2.14 Parser_getChar() . . . 356
9.15.2.15 Parser_getNameSpace() . . . 357
9.15.2.16 Parser_getNextNode() . . . 357
9.15.2.17 Parser_getNextToken() . . . 357
9.15.2.18 Parser_hasDefaultNamespace() . . . 358
9.15.2.19 Parser_init() . . . 358
9.15.2.20 Parser_intToUTF8() . . . 358
9.15.2.21 Parser_isCharInTable() . . . 359
9.15.2.22 Parser_isNameChar() . . . 359
9.15.2.23 Parser_isValidEndElement() . . . 359
9.15.2.24 Parser_isValidXmlName() . . . 360
9.15.2.25 Parser_isXmlChar() . . . 360
9.15.2.26 Parser_LoadDocument() . . . 360
9.15.2.27 Parser_parseDocument() . . . 361
9.15.2.28 Parser_popElement() . . . 361
9.15.2.29 Parser_processAttribute() . . . 362
9.15.2.30 Parser_processAttributeName() . . . 362
9.15.2.31 Parser_processCDSect() . . . 362
9.15.2.32 Parser_processContent() . . . 363

9.15.2.33 Parser_processElementName() . . . 363
9.15.2.34 Parser_processETag() . . . 364
9.15.2.35 Parser_processSTag() . . . 364
9.15.2.36 Parser_pushElement() . . . 364
9.15.2.37 Parser_readFileOrBuffer() . . . 365
9.15.2.38 Parser_setElementNamespace() . . . 365
9.15.2.39 Parser_setErrorChar() . . . 366
9.15.2.40 Parser_setLastElem() . . . 366
9.15.2.41 Parser_setNodePrefixAndLocalName() . . . 366
9.15.2.42 Parser_skipBom() . . . 367
9.15.2.43 Parser_skipComment() . . . 367
9.15.2.44 Parser_skipDocType() . . . 367
9.15.2.45 Parser_skipMisc() . . . 367
9.15.2.46 Parser_skipPI() . . . 368
9.15.2.47 Parser_skipProlog() . . . 368
9.15.2.48 Parser_skipString() . . . 368
9.15.2.49 Parser_skipWhiteSpaces() . . . 369
9.15.2.50 Parser_skipXMLDecl() . . . 369
9.15.2.51 Parser_UTF8ToInt() . . . 369
9.15.2.52 Parser_xmlNamespace() . . . 370
9.15.2.53 safe_strdup() . . . 370
9.15.3 Variable Documentation . . . 370
9.15.3.1 Letter . . . 370
9.15.3.2 NameChar . . . 371
9.16 /build/reproducible-path/pupnp-1.14.20/ixml/src/namedNodeMap.c File Reference . . . 371
9.16.1 Function Documentation . . . 372
9.16.1.1 ixmlNamedNodeMap_addToNamedNodeMap() . . . 372
9.16.1.2 ixmlNamedNodeMap_getItemNumber() . . . 372
9.16.1.3 ixmlNamedNodeMap_init() . . . 373
9.17 /build/reproducible-path/pupnp-1.14.20/ixml/src/node.c File Reference . . . 373
9.17.1 Function Documentation . . . 375
9.17.1.1 ixmlNode_allowChildren() . . . 375
9.17.1.2 ixmlNode_cloneAttr() . . . 376
9.17.1.3 ixmlNode_cloneAttrDirect() . . . 376
9.17.1.4 ixmlNode_cloneCDATASect() . . . 377
9.17.1.5 ixmlNode_cloneElement() . . . 377
9.17.1.6 ixmlNode_cloneNodeTree() . . . 377
9.17.1.7 ixmlNode_cloneNodeTreeRecursive() . . . 378
9.17.1.8 ixmlNode_cloneTextNode() . . . 378
9.17.1.9 ixmlNode_compare() . . . 379
9.17.1.10 ixmlNode_freeSingleNode() . . . 379
9.17.1.11 ixmlNode_getElementsByTagName() . . . 379

9.17.1.12 ixmlNode_getElementsByTagNameNS() . . . 380
9.17.1.13 ixmlNode_getElementsByTagNameNSRecursive() . . . 380
9.17.1.14 ixmlNode_getElementsByTagNameRecursive() . . . 381
9.17.1.15 ixmlNode_init() . . . 381
9.17.1.16 ixmlNode_isAncestor() . . . 381
9.17.1.17 ixmlNode_isParent() . . . 382
9.17.1.18 ixmlNode_newDoc() . . . 382
9.17.1.19 ixmlNode_setLocalName() . . . 382
9.17.1.20 ixmlNode_setNamespaceURI() . . . 383
9.17.1.21 ixmlNode_setNodeName() . . . 383
9.17.1.22 ixmlNode_setNodeProperties() . . . 383
9.17.1.23 ixmlNode_setPrefix() . . . 384
9.17.1.24 ixmlNode_setSiblingNodesParent() . . . 384
9.18 /build/reproducible-path/pupnp-1.14.20/ixml/src/nodeList.c File Reference . . . 384
9.18.1 Function Documentation . . . 385
9.18.1.1 ixmlNodeList_addToNodeList() . . . 385
9.18.1.2 ixmlNodeList_init() . . . 386
9.19 generator.h . . . 386
9.20 /build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.c File Reference . . . 387
9.20.1 Detailed Description . . . 388
9.20.2 Function Documentation . . . 388
9.20.2.1 TestClass_add_to_list_TheList() . . . 388
9.20.2.2 TestClass_assign() . . . 389
9.20.2.3 TestClass_clear_TheBuffer() . . . 389
9.20.2.4 TestClass_clear_TheString() . . . 389
9.20.2.5 TestClass_delete() . . . 389
9.20.2.6 TestClass_dup() . . . 389
9.20.2.7 TestClass_get_TheActionComplete() . . . 390
9.20.2.8 TestClass_get_TheBuffer() . . . 390
9.20.2.9 TestClass_get_TheCharPointer() . . . 390
9.20.2.10 TestClass_get_TheDomString() . . . 390
9.20.2.11 TestClass_get_TheDomString_cstr() . . . 390
9.20.2.12 TestClass_get_TheInteger() . . . 390
9.20.2.13 TestClass_get_TheList() . . . 391
9.20.2.14 TestClass_get_TheString() . . . 391
9.20.2.15 TestClass_get_TheString_cstr() . . . 391
9.20.2.16 TestClass_get_TheString_Length() . . . 391
9.20.2.17 TestClass_get_TheUnsignedLong() . . . 391
9.20.2.18 TestClass_new() . . . 392
9.20.2.19 TestClass_set_TheActionComplete() . . . 392
9.20.2.20 TestClass_set_TheBuffer() . . . 392
9.20.2.21 TestClass_set_TheCharPointer() . . . 392

9.20.2.22 TestClass_set_TheDomString() . . . 392
9.20.2.23 TestClass_set_TheInteger() . . . 393
9.20.2.24 TestClass_set_TheList() . . . 393
9.20.2.25 TestClass_set_TheString() . . . 393
9.20.2.26 TestClass_set_TheUnsignedLong() . . . 393
9.20.2.27 TestClass_strcpy_TheString() . . . 393
9.20.2.28 TestClass_strncpy_TheString() . . . 394
9.21 /build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.h File Reference . . . 394
9.21.1 Detailed Description . . . 396
9.21.2 Typedef Documentation . . . 396
9.21.2.1 TestClass . . . 396
9.21.3 Function Documentation . . . 396
9.21.3.1 TestClass_add_to_list_TheList() . . . 396
9.21.3.2 TestClass_assign() . . . 396
9.21.3.3 TestClass_clear_TheBuffer() . . . 396
9.21.3.4 TestClass_clear_TheString() . . . 397
9.21.3.5 TestClass_delete() . . . 397
9.21.3.6 TestClass_dup() . . . 397
9.21.3.7 TestClass_get_TheActionComplete() . . . 397
9.21.3.8 TestClass_get_TheBuffer() . . . 397
9.21.3.9 TestClass_get_TheCharPointer() . . . 397
9.21.3.10 TestClass_get_TheDomString() . . . 398
9.21.3.11 TestClass_get_TheDomString_cstr() . . . 398
9.21.3.12 TestClass_get_TheInteger() . . . 398
9.21.3.13 TestClass_get_TheList() . . . 398
9.21.3.14 TestClass_get_TheString() . . . 398
9.21.3.15 TestClass_get_TheString_cstr() . . . 398
9.21.3.16 TestClass_get_TheString_Length() . . . 399
9.21.3.17 TestClass_get_TheUnsignedLong() . . . 399
9.21.3.18 TestClass_new() . . . 399
9.21.3.19 TestClass_set_TheActionComplete() . . . 399
9.21.3.20 TestClass_set_TheBuffer() . . . 399
9.21.3.21 TestClass_set_TheCharPointer() . . . 400
9.21.3.22 TestClass_set_TheDomString() . . . 400
9.21.3.23 TestClass_set_TheInteger() . . . 400
9.21.3.24 TestClass_set_TheList() . . . 400
9.21.3.25 TestClass_set_TheString() . . . 400
9.21.3.26 TestClass_set_TheUnsignedLong() . . . 401
9.21.3.27 TestClass_strcpy_TheString() . . . 401
9.21.3.28 TestClass_strncpy_TheString() . . . 401
9.22 TestClass.h . . . 401
9.23 TheStruct.h . . . 402

9.24 /build/reproducible-path/pupnp-1.14.20/upnp/inc/Callback.h File Reference . . . 402
9.24.1 Typedef Documentation . . . 403
9.24.1.1 Upnp_FunPtr . . . 403
9.24.2 Enumeration Type Documentation . . . 403
9.24.2.1 Upnp_EventType_e . . . 403
9.25 Callback.h . . . 405
9.26 /build/reproducible-path/pupnp-1.14.20/upnp/inc/ithread.h File Reference . . . 406
9.27 ithread.h . . . 408
9.28 list.h . . . 418
9.29 /build/reproducible-path/pupnp-1.14.20/upnp/inc/upnp.h File Reference . . . 419
9.30 upnp.h . . . 426
9.31 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionComplete.h File Reference . . . 435
9.31.1 Detailed Description . . . 436
9.31.2 Typedef Documentation . . . 436
9.31.2.1 UpnpActionComplete . . . 436
9.31.3 Function Documentation . . . 436
9.31.3.1 UpnpActionComplete_assign() . . . 436
9.31.3.2 UpnpActionComplete_clear_CtrlUrl() . . . 437
9.31.3.3 UpnpActionComplete_delete() . . . 437
9.31.3.4 UpnpActionComplete_dup() . . . 437
9.31.3.5 UpnpActionComplete_get_ActionRequest() . . . 437
9.31.3.6 UpnpActionComplete_get_ActionResult() . . . 437
9.31.3.7 UpnpActionComplete_get_CtrlUrl() . . . 438
9.31.3.8 UpnpActionComplete_get_CtrlUrl_cstr() . . . 438
9.31.3.9 UpnpActionComplete_get_CtrlUrl_Length() . . . 438
9.31.3.10 UpnpActionComplete_get_ErrCode() . . . 438
9.31.3.11 UpnpActionComplete_new() . . . 438
9.31.3.12 UpnpActionComplete_set_ActionRequest() . . . 439
9.31.3.13 UpnpActionComplete_set_ActionResult() . . . 439
9.31.3.14 UpnpActionComplete_set_CtrlUrl() . . . 439
9.31.3.15 UpnpActionComplete_set_ErrCode() . . . 439
9.31.3.16 UpnpActionComplete_strcpy_CtrlUrl() . . . 439
9.31.3.17 UpnpActionComplete_strncpy_CtrlUrl() . . . 440
9.32 UpnpActionComplete.h . . . 440
9.33 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionRequest.h File Reference . . . 441
9.33.1 Detailed Description . . . 442
9.33.2 Typedef Documentation . . . 443
9.33.2.1 UpnpActionRequest . . . 443
9.33.3 Function Documentation . . . 443
9.33.3.1 UpnpActionRequest_assign() . . . 443
9.33.3.2 UpnpActionRequest_clear_ActionName() . . . 443
9.33.3.3 UpnpActionRequest_clear_CtrlPtIPAddr() . . . 443

9.33.3.4 UpnpActionRequest_clear_DevUDN() . . . 443
9.33.3.5 UpnpActionRequest_clear_ErrStr() . . . 444
9.33.3.6 UpnpActionRequest_clear_Os() . . . 444
9.33.3.7 UpnpActionRequest_clear_ServiceID() . . . 444
9.33.3.8 UpnpActionRequest_delete() . . . 444
9.33.3.9 UpnpActionRequest_dup() . . . 444
9.33.3.10 UpnpActionRequest_get_ActionName() . . . 445
9.33.3.11 UpnpActionRequest_get_ActionName_cstr() . . . 445
9.33.3.12 UpnpActionRequest_get_ActionName_Length() . . . 445
9.33.3.13 UpnpActionRequest_get_ActionRequest() . . . 445
9.33.3.14 UpnpActionRequest_get_ActionResult() . . . 445
9.33.3.15 UpnpActionRequest_get_CtrlPtIPAddr() . . . 446
9.33.3.16 UpnpActionRequest_get_DevUDN() . . . 446
9.33.3.17 UpnpActionRequest_get_DevUDN_cstr() . . . 446
9.33.3.18 UpnpActionRequest_get_DevUDN_Length() . . . 446
9.33.3.19 UpnpActionRequest_get_ErrCode() . . . 446
9.33.3.20 UpnpActionRequest_get_ErrStr() . . . 447
9.33.3.21 UpnpActionRequest_get_ErrStr_cstr() . . . 447
9.33.3.22 UpnpActionRequest_get_ErrStr_Length() . . . 447
9.33.3.23 UpnpActionRequest_get_Os() . . . 447
9.33.3.24 UpnpActionRequest_get_Os_cstr() . . . 447
9.33.3.25 UpnpActionRequest_get_Os_Length() . . . 448
9.33.3.26 UpnpActionRequest_get_ServiceID() . . . 448
9.33.3.27 UpnpActionRequest_get_ServiceID_cstr() . . . 448
9.33.3.28 UpnpActionRequest_get_ServiceID_Length() . . . 448
9.33.3.29 UpnpActionRequest_get_SoapHeader() . . . 448
9.33.3.30 UpnpActionRequest_get_Socket() . . . 449
9.33.3.31 UpnpActionRequest_new() . . . 449
9.33.3.32 UpnpActionRequest_set_ActionName() . . . 449
9.33.3.33 UpnpActionRequest_set_ActionRequest() . . . 449
9.33.3.34 UpnpActionRequest_set_ActionResult() . . . 449
9.33.3.35 UpnpActionRequest_set_CtrlPtIPAddr() . . . 450
9.33.3.36 UpnpActionRequest_set_DevUDN() . . . 450
9.33.3.37 UpnpActionRequest_set_ErrCode() . . . 450
9.33.3.38 UpnpActionRequest_set_ErrStr() . . . 450
9.33.3.39 UpnpActionRequest_set_Os() . . . 450
9.33.3.40 UpnpActionRequest_set_ServiceID() . . . 451
9.33.3.41 UpnpActionRequest_set_SoapHeader() . . . 451
9.33.3.42 UpnpActionRequest_set_Socket() . . . 451
9.33.3.43 UpnpActionRequest_strcpy_ActionName() . . . 451
9.33.3.44 UpnpActionRequest_strcpy_DevUDN() . . . 451
9.33.3.45 UpnpActionRequest_strcpy_ErrStr() . . . 452

9.33.3.46 UpnpActionRequest_strcpy_Os() . . . 452
9.33.3.47 UpnpActionRequest_strcpy_ServiceID() . . . 452
9.33.3.48 UpnpActionRequest_strncpy_ActionName() . . . 452
9.33.3.49 UpnpActionRequest_strncpy_DevUDN() . . . 452
9.33.3.50 UpnpActionRequest_strncpy_ErrStr() . . . 453
9.33.3.51 UpnpActionRequest_strncpy_Os() . . . 453
9.33.3.52 UpnpActionRequest_strncpy_ServiceID() . . . 453
9.34 UpnpActionRequest.h . . . 453
9.35 /build/reproducible-path/pupnp-1.14.20/upnp/inc/upnpdebug.h File Reference . . . 455
9.35.1 Macro Definition Documentation . . . 456
9.35.1.1 UPNP_DEFAULT_LOG_LEVEL . . . 456
9.35.2 Function Documentation . . . 456
9.35.2.1 UpnpCloseLog() . . . 456
9.35.2.2 UpnpGetDebugFile() . . . 457
9.35.2.3 UpnpInitLog() . . . 457
9.35.2.4 UpnpPrintf() . . . 457
9.35.2.5 UpnpSetLogFileNames() . . . 458
9.35.2.6 UpnpSetLogLevel() . . . 458
9.36 upnpdebug.h . . . 459
9.37 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpDiscovery.h File Reference . . . 461
9.37.1 Detailed Description . . . 463
9.37.2 Typedef Documentation . . . 463
9.37.2.1 UpnpDiscovery . . . 463
9.37.3 Function Documentation . . . 463
9.37.3.1 UpnpDiscovery_assign() . . . 463
9.37.3.2 UpnpDiscovery_clear_Date() . . . 463
9.37.3.3 UpnpDiscovery_clear_DestAddr() . . . 464
9.37.3.4 UpnpDiscovery_clear_DeviceID() . . . 464
9.37.3.5 UpnpDiscovery_clear_DeviceType() . . . 464
9.37.3.6 UpnpDiscovery_clear_Ext() . . . 464
9.37.3.7 UpnpDiscovery_clear_Location() . . . 464
9.37.3.8 UpnpDiscovery_clear_Os() . . . 464
9.37.3.9 UpnpDiscovery_clear_ServiceType() . . . 465
9.37.3.10 UpnpDiscovery_clear_ServiceVer() . . . 465
9.37.3.11 UpnpDiscovery_delete() . . . 465
9.37.3.12 UpnpDiscovery_dup() . . . 465
9.37.3.13 UpnpDiscovery_get_Date() . . . 465
9.37.3.14 UpnpDiscovery_get_Date_cstr() . . . 466
9.37.3.15 UpnpDiscovery_get_Date_Length() . . . 466
9.37.3.16 UpnpDiscovery_get_DestAddr() . . . 466
9.37.3.17 UpnpDiscovery_get_DeviceID() . . . 466
9.37.3.18 UpnpDiscovery_get_DeviceID_cstr() . . . 466

9.37.3.19 UpnpDiscovery_get_DeviceID_Length() . . . 467
9.37.3.20 UpnpDiscovery_get_DeviceType() . . . 467
9.37.3.21 UpnpDiscovery_get_DeviceType_cstr() . . . 467
9.37.3.22 UpnpDiscovery_get_DeviceType_Length() . . . 467
9.37.3.23 UpnpDiscovery_get_ErrCode() . . . 467
9.37.3.24 UpnpDiscovery_get_Expires() . . . 468
9.37.3.25 UpnpDiscovery_get_Ext() . . . 468
9.37.3.26 UpnpDiscovery_get_Ext_cstr() . . . 468
9.37.3.27 UpnpDiscovery_get_Ext_Length() . . . 468
9.37.3.28 UpnpDiscovery_get_Location() . . . 468
9.37.3.29 UpnpDiscovery_get_Location_cstr() . . . 469
9.37.3.30 UpnpDiscovery_get_Location_Length() . . . 469
9.37.3.31 UpnpDiscovery_get_Os() . . . 469
9.37.3.32 UpnpDiscovery_get_Os_cstr() . . . 469
9.37.3.33 UpnpDiscovery_get_Os_Length() . . . 469
9.37.3.34 UpnpDiscovery_get_ServiceType() . . . 470
9.37.3.35 UpnpDiscovery_get_ServiceType_cstr() . . . 470
9.37.3.36 UpnpDiscovery_get_ServiceType_Length() . . . 470
9.37.3.37 UpnpDiscovery_get_ServiceVer() . . . 470
9.37.3.38 UpnpDiscovery_get_ServiceVer_cstr() . . . 470
9.37.3.39 UpnpDiscovery_get_ServiceVer_Length() . . . 471
9.37.3.40 UpnpDiscovery_new() . . . 471
9.37.3.41 UpnpDiscovery_set_Date() . . . 471
9.37.3.42 UpnpDiscovery_set_DestAddr() . . . 471
9.37.3.43 UpnpDiscovery_set_DeviceID() . . . 471
9.37.3.44 UpnpDiscovery_set_DeviceType() . . . 472
9.37.3.45 UpnpDiscovery_set_ErrCode() . . . 472
9.37.3.46 UpnpDiscovery_set_Expires() . . . 472
9.37.3.47 UpnpDiscovery_set_Ext() . . . 472
9.37.3.48 UpnpDiscovery_set_Location() . . . 472
9.37.3.49 UpnpDiscovery_set_Os() . . . 473
9.37.3.50 UpnpDiscovery_set_ServiceType() . . . 473
9.37.3.51 UpnpDiscovery_set_ServiceVer() . . . 473
9.37.3.52 UpnpDiscovery_strcpy_Date() . . . 473
9.37.3.53 UpnpDiscovery_strcpy_DeviceID() . . . 474
9.37.3.54 UpnpDiscovery_strcpy_DeviceType() . . . 474
9.37.3.55 UpnpDiscovery_strcpy_Ext() . . . 474
9.37.3.56 UpnpDiscovery_strcpy_Location() . . . 474
9.37.3.57 UpnpDiscovery_strcpy_Os() . . . 474
9.37.3.58 UpnpDiscovery_strcpy_ServiceType() . . . 475
9.37.3.59 UpnpDiscovery_strcpy_ServiceVer() . . . 475
9.37.3.60 UpnpDiscovery_strncpy_Date() . . . 475

9.37.3.61 UpnpDiscovery_strncpy_DeviceID() . . . 475
9.37.3.62 UpnpDiscovery_strncpy_DeviceType() . . . 475
9.37.3.63 UpnpDiscovery_strncpy_Ext() . . . 476
9.37.3.64 UpnpDiscovery_strncpy_Location() . . . 476
9.37.3.65 UpnpDiscovery_strncpy_Os() . . . 476
9.37.3.66 UpnpDiscovery_strncpy_ServiceType() . . . 476
9.37.3.67 UpnpDiscovery_strncpy_ServiceVer() . . . 477
9.38 UpnpDiscovery.h . . . 477
9.39 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEvent.h File Reference . . . 479
9.39.1 Detailed Description . . . 480
9.39.2 Typedef Documentation . . . 480
9.39.2.1 UpnpEvent . . . 480
9.39.3 Function Documentation . . . 480
9.39.3.1 UpnpEvent_assign() . . . 480
9.39.3.2 UpnpEvent_clear_SID() . . . 480
9.39.3.3 UpnpEvent_delete() . . . 481
9.39.3.4 UpnpEvent_dup() . . . 481
9.39.3.5 UpnpEvent_get_ChangedVariables() . . . 481
9.39.3.6 UpnpEvent_get_EventKey() . . . 481
9.39.3.7 UpnpEvent_get_SID() . . . 481
9.39.3.8 UpnpEvent_get_SID_cstr() . . . 482
9.39.3.9 UpnpEvent_get_SID_Length() . . . 482
9.39.3.10 UpnpEvent_new() . . . 482
9.39.3.11 UpnpEvent_set_ChangedVariables() . . . 482
9.39.3.12 UpnpEvent_set_EventKey() . . . 482
9.39.3.13 UpnpEvent_set_SID() . . . 483
9.39.3.14 UpnpEvent_strcpy_SID() . . . 483
9.39.3.15 UpnpEvent_strncpy_SID() . . . 483
9.40 UpnpEvent.h . . . 483
9.41 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEventSubscribe.h File Reference . . . 484
9.41.1 Detailed Description . . . 485
9.41.2 Typedef Documentation . . . 485
9.41.2.1 UpnpEventSubscribe . . . 485
9.41.3 Function Documentation . . . 486
9.41.3.1 UpnpEventSubscribe_assign() . . . 486
9.41.3.2 UpnpEventSubscribe_clear_PublisherUrl() . . . 486
9.41.3.3 UpnpEventSubscribe_clear_SID() . . . 486
9.41.3.4 UpnpEventSubscribe_delete() . . . 486
9.41.3.5 UpnpEventSubscribe_dup() . . . 486
9.41.3.6 UpnpEventSubscribe_get_ErrCode() . . . 487
9.41.3.7 UpnpEventSubscribe_get_PublisherUrl() . . . 487
9.41.3.8 UpnpEventSubscribe_get_PublisherUrl_cstr() . . . 487

9.41.3.9 UpnpEventSubscribe_get_PublisherUrl_Length() . . . 487
9.41.3.10 UpnpEventSubscribe_get_SID() . . . 487
9.41.3.11 UpnpEventSubscribe_get_SID_cstr() . . . 488
9.41.3.12 UpnpEventSubscribe_get_SID_Length() . . . 488
9.41.3.13 UpnpEventSubscribe_get_TimeOut() . . . 488
9.41.3.14 UpnpEventSubscribe_new() . . . 488
9.41.3.15 UpnpEventSubscribe_set_ErrCode() . . . 488
9.41.3.16 UpnpEventSubscribe_set_PublisherUrl() . . . 489
9.41.3.17 UpnpEventSubscribe_set_SID() . . . 489
9.41.3.18 UpnpEventSubscribe_set_TimeOut() . . . 489
9.41.3.19 UpnpEventSubscribe_strcpy_PublisherUrl() . . . 489
9.41.3.20 UpnpEventSubscribe_strcpy_SID() . . . 490
9.41.3.21 UpnpEventSubscribe_strncpy_PublisherUrl() . . . 490
9.41.3.22 UpnpEventSubscribe_strncpy_SID() . . . 490
9.42 UpnpEventSubscribe.h . . . 490
9.43 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpExtraHeaders.h File Reference . . . 491
9.43.1 Detailed Description . . . 493
9.43.2 Typedef Documentation . . . 493
9.43.2.1 UpnpExtraHeaders . . . 493
9.43.3 Function Documentation . . . 493
9.43.3.1 UpnpExtraHeaders_add_to_list_node() . . . 493
9.43.3.2 UpnpExtraHeaders_assign() . . . 494
9.43.3.3 UpnpExtraHeaders_clear_name() . . . 494
9.43.3.4 UpnpExtraHeaders_clear_value() . . . 494
9.43.3.5 UpnpExtraHeaders_delete() . . . 494
9.43.3.6 UpnpExtraHeaders_dup() . . . 494
9.43.3.7 UpnpExtraHeaders_get_name() . . . 495
9.43.3.8 UpnpExtraHeaders_get_name_cstr() . . . 495
9.43.3.9 UpnpExtraHeaders_get_name_Length() . . . 495
9.43.3.10 UpnpExtraHeaders_get_node() . . . 495
9.43.3.11 UpnpExtraHeaders_get_resp() . . . 495
9.43.3.12 UpnpExtraHeaders_get_resp_cstr() . . . 495
9.43.3.13 UpnpExtraHeaders_get_value() . . . 496
9.43.3.14 UpnpExtraHeaders_get_value_cstr() . . . 496
9.43.3.15 UpnpExtraHeaders_get_value_Length() . . . 496
9.43.3.16 UpnpExtraHeaders_new() . . . 496
9.43.3.17 UpnpExtraHeaders_set_name() . . . 496
9.43.3.18 UpnpExtraHeaders_set_node() . . . 497
9.43.3.19 UpnpExtraHeaders_set_resp() . . . 497
9.43.3.20 UpnpExtraHeaders_set_value() . . . 497
9.43.3.21 UpnpExtraHeaders_strcpy_name() . . . 497
9.43.3.22 UpnpExtraHeaders_strcpy_value() . . . 497

9.43.3.23 UpnpExtraHeaders_strncpy_name() . . . 498
9.43.3.24 UpnpExtraHeaders_strncpy_value() . . . 498
9.44 UpnpExtraHeaders.h . . . 498
9.45 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpFileInfo.h File Reference . . . 499
9.45.1 Detailed Description . . . 500
9.45.2 Typedef Documentation . . . 501
9.45.2.1 UpnpFileInfo . . . 501
9.45.3 Function Documentation . . . 501
9.45.3.1 UpnpFileInfo_add_to_list_ExtraHeadersList() . . . 501
9.45.3.2 UpnpFileInfo_assign() . . . 501
9.45.3.3 UpnpFileInfo_clear_CtrlPtIPAddr() . . . 501
9.45.3.4 UpnpFileInfo_clear_Os() . . . 501
9.45.3.5 UpnpFileInfo_delete() . . . 502
9.45.3.6 UpnpFileInfo_dup() . . . 502
9.45.3.7 UpnpFileInfo_get_ContentType() . . . 502
9.45.3.8 UpnpFileInfo_get_ContentType_cstr() . . . 502
9.45.3.9 UpnpFileInfo_get_CtrlPtIPAddr() . . . 502
9.45.3.10 UpnpFileInfo_get_ExtraHeadersList() . . . 503
9.45.3.11 UpnpFileInfo_get_FileLength() . . . 503
9.45.3.12 UpnpFileInfo_get_IsDirectory() . . . 503
9.45.3.13 UpnpFileInfo_get_IsReadable() . . . 503
9.45.3.14 UpnpFileInfo_get_LastModified() . . . 503
9.45.3.15 UpnpFileInfo_get_Os() . . . 503
9.45.3.16 UpnpFileInfo_get_Os_cstr() . . . 504
9.45.3.17 UpnpFileInfo_get_Os_Length() . . . 504
9.45.3.18 UpnpFileInfo_new() . . . 504
9.45.3.19 UpnpFileInfo_set_ContentType() . . . 504
9.45.3.20 UpnpFileInfo_set_CtrlPtIPAddr() . . . 504
9.45.3.21 UpnpFileInfo_set_ExtraHeadersList() . . . 505
9.45.3.22 UpnpFileInfo_set_FileLength() . . . 505
9.45.3.23 UpnpFileInfo_set_IsDirectory() . . . 505
9.45.3.24 UpnpFileInfo_set_IsReadable() . . . 505
9.45.3.25 UpnpFileInfo_set_LastModified() . . . 505
9.45.3.26 UpnpFileInfo_set_Os() . . . 506
9.45.3.27 UpnpFileInfo_strcpy_Os() . . . 506
9.45.3.28 UpnpFileInfo_strncpy_Os() . . . 506
9.46 UpnpFileInfo.h . . . 506
9.47 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpGlobal.h File Reference . . . 507
9.47.1 Detailed Description . . . 508
9.47.2 Macro Definition Documentation . . . 508
9.47.2.1 PRIzd . . . 508
9.47.2.2 UPNP_EXPORT_SPEC . . . 508

9.47.2.3 UPNP_INLINE . . . 508
9.48 UpnpGlobal.h . . . 509
9.49 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpInet.h File Reference . . . 510
9.49.1 Detailed Description . . . 511
9.50 UpnpInet.h . . . 511
9.51 UpnpIntTypes.h . . . 512
9.52 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarComplete.h File Reference . . . 512
9.52.1 Detailed Description . . . 513
9.52.2 Typedef Documentation . . . 513
9.52.2.1 UpnpStateVarComplete . . . 513
9.52.3 Function Documentation . . . 514
9.52.3.1 UpnpStateVarComplete_assign() . . . 514
9.52.3.2 UpnpStateVarComplete_clear_CtrlUrl() . . . 514
9.52.3.3 UpnpStateVarComplete_clear_StateVarName() . . . 514
9.52.3.4 UpnpStateVarComplete_delete() . . . 514
9.52.3.5 UpnpStateVarComplete_dup() . . . 514
9.52.3.6 UpnpStateVarComplete_get_CtrlUrl() . . . 515
9.52.3.7 UpnpStateVarComplete_get_CtrlUrl_cstr() . . . 515
9.52.3.8 UpnpStateVarComplete_get_CtrlUrl_Length() . . . 515
9.52.3.9 UpnpStateVarComplete_get_CurrentVal() . . . 515
9.52.3.10 UpnpStateVarComplete_get_CurrentVal_cstr() . . . 515
9.52.3.11 UpnpStateVarComplete_get_ErrCode() . . . 516
9.52.3.12 UpnpStateVarComplete_get_StateVarName() . . . 516
9.52.3.13 UpnpStateVarComplete_get_StateVarName_cstr() . . . 516
9.52.3.14 UpnpStateVarComplete_get_StateVarName_Length() . . . 516
9.52.3.15 UpnpStateVarComplete_new() . . . 516
9.52.3.16 UpnpStateVarComplete_set_CtrlUrl() . . . 517
9.52.3.17 UpnpStateVarComplete_set_CurrentVal() . . . 517
9.52.3.18 UpnpStateVarComplete_set_ErrCode() . . . 517
9.52.3.19 UpnpStateVarComplete_set_StateVarName() . . . 517
9.52.3.20 UpnpStateVarComplete_strcpy_CtrlUrl() . . . 518
9.52.3.21 UpnpStateVarComplete_strcpy_StateVarName() . . . 518
9.52.3.22 UpnpStateVarComplete_strncpy_CtrlUrl() . . . 518
9.52.3.23 UpnpStateVarComplete_strncpy_StateVarName() . . . 518
9.53 UpnpStateVarComplete.h . . . 519
9.54 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarRequest.h File Reference . . . 519
9.54.1 Detailed Description . . . 521
9.54.2 Typedef Documentation . . . 521
9.54.2.1 UpnpStateVarRequest . . . 521
9.54.3 Function Documentation . . . 522
9.54.3.1 UpnpStateVarRequest_assign() . . . 522
9.54.3.2 UpnpStateVarRequest_clear_CtrlPtIPAddr() . . . 522

9.54.3.3 UpnpStateVarRequest_clear_DevUDN() . . . 522
9.54.3.4 UpnpStateVarRequest_clear_ErrStr() . . . 522
9.54.3.5 UpnpStateVarRequest_clear_ServiceID() . . . 522
9.54.3.6 UpnpStateVarRequest_clear_StateVarName() . . . 523
9.54.3.7 UpnpStateVarRequest_delete() . . . 523
9.54.3.8 UpnpStateVarRequest_dup() . . . 523
9.54.3.9 UpnpStateVarRequest_get_CtrlPtIPAddr() . . . 523
9.54.3.10 UpnpStateVarRequest_get_CurrentVal() . . . 523
9.54.3.11 UpnpStateVarRequest_get_CurrentVal_cstr() . . . 524
9.54.3.12 UpnpStateVarRequest_get_DevUDN() . . . 524
9.54.3.13 UpnpStateVarRequest_get_DevUDN_cstr() . . . 524
9.54.3.14 UpnpStateVarRequest_get_DevUDN_Length() . . . 524
9.54.3.15 UpnpStateVarRequest_get_ErrCode() . . . 524
9.54.3.16 UpnpStateVarRequest_get_ErrStr() . . . 525
9.54.3.17 UpnpStateVarRequest_get_ErrStr_cstr() . . . 525
9.54.3.18 UpnpStateVarRequest_get_ErrStr_Length() . . . 525
9.54.3.19 UpnpStateVarRequest_get_ServiceID() . . . 525
9.54.3.20 UpnpStateVarRequest_get_ServiceID_cstr() . . . 525
9.54.3.21 UpnpStateVarRequest_get_ServiceID_Length() . . . 526
9.54.3.22 UpnpStateVarRequest_get_Socket() . . . 526
9.54.3.23 UpnpStateVarRequest_get_StateVarName() . . . 526
9.54.3.24 UpnpStateVarRequest_get_StateVarName_cstr() . . . 526
9.54.3.25 UpnpStateVarRequest_get_StateVarName_Length() . . . 526
9.54.3.26 UpnpStateVarRequest_new() . . . 527
9.54.3.27 UpnpStateVarRequest_set_CtrlPtIPAddr() . . . 527
9.54.3.28 UpnpStateVarRequest_set_CurrentVal() . . . 527
9.54.3.29 UpnpStateVarRequest_set_DevUDN() . . . 527
9.54.3.30 UpnpStateVarRequest_set_ErrCode() . . . 527
9.54.3.31 UpnpStateVarRequest_set_ErrStr() . . . 528
9.54.3.32 UpnpStateVarRequest_set_ServiceID() . . . 528
9.54.3.33 UpnpStateVarRequest_set_Socket() . . . 528
9.54.3.34 UpnpStateVarRequest_set_StateVarName() . . . 528
9.54.3.35 UpnpStateVarRequest_strcpy_DevUDN() . . . 529
9.54.3.36 UpnpStateVarRequest_strcpy_ErrStr() . . . 529
9.54.3.37 UpnpStateVarRequest_strcpy_ServiceID() . . . 529
9.54.3.38 UpnpStateVarRequest_strcpy_StateVarName() . . . 529
9.54.3.39 UpnpStateVarRequest_strncpy_DevUDN() . . . 530
9.54.3.40 UpnpStateVarRequest_strncpy_ErrStr() . . . 530
9.54.3.41 UpnpStateVarRequest_strncpy_ServiceID() . . . 530
9.54.3.42 UpnpStateVarRequest_strncpy_StateVarName() . . . 530
9.55 UpnpStateVarRequest.h . . . 531
9.56 UpnpStdInt.h . . . 532

9.57 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpString.h File Reference 532
9.57.1 Detailed Description 534
9.58 UpnpString.h 534
9.59 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpSubscriptionRequest.h File Reference 535
9.59.1 Detailed Description 536
9.59.2 Typedef Documentation 536
9.59.2.1 UpnpSubscriptionRequest 536
9.59.3 Function Documentation 537
9.59.3.1 UpnpSubscriptionRequest_assign() 537
9.59.3.2 UpnpSubscriptionRequest_clear_ServiceId() 537
9.59.3.3 UpnpSubscriptionRequest_clear_SID() 537
9.59.3.4 UpnpSubscriptionRequest_clear_UDN() 537
9.59.3.5 UpnpSubscriptionRequest_delete() 537
9.59.3.6 UpnpSubscriptionRequest_dup() 538
9.59.3.7 UpnpSubscriptionRequest_get_ServiceId() 538
9.59.3.8 UpnpSubscriptionRequest_get_ServiceId_cstr() 538
9.59.3.9 UpnpSubscriptionRequest_get_ServiceId_Length() 538
9.59.3.10 UpnpSubscriptionRequest_get_SID() 538
9.59.3.11 UpnpSubscriptionRequest_get_SID_cstr() 539
9.59.3.12 UpnpSubscriptionRequest_get_SID_Length() 539
9.59.3.13 UpnpSubscriptionRequest_get_UDN() 539
9.59.3.14 UpnpSubscriptionRequest_get_UDN_cstr() 539
9.59.3.15 UpnpSubscriptionRequest_get_UDN_Length() 539
9.59.3.16 UpnpSubscriptionRequest_new() 540
9.59.3.17 UpnpSubscriptionRequest_set_ServiceId() 540
9.59.3.18 UpnpSubscriptionRequest_set_SID() 540
9.59.3.19 UpnpSubscriptionRequest_set_UDN() 540
9.59.3.20 UpnpSubscriptionRequest_strcpy_ServiceId() 541
9.59.3.21 UpnpSubscriptionRequest_strcpy_SID() 541
9.59.3.22 UpnpSubscriptionRequest_strcpy_UDN() 541
9.59.3.23 UpnpSubscriptionRequest_strncpy_ServiceId() 541
9.59.3.24 UpnpSubscriptionRequest_strncpy_SID() 542
9.59.3.25 UpnpSubscriptionRequest_strncpy_UDN() 542
9.60 UpnpSubscriptionRequest.h 542
9.61 /build/reproducible-path/pupnp-1.14.20/upnp/inc/upnptools.h File Reference 543
9.62 upnptools.h 545
9.63 UpnpUniStd.h 546
9.64 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util.c File Reference 546
9.65 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util.h File Reference 547
9.66 sample_util.h 549
9.67 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.c File Reference 551
9.68 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.h File Reference 553

9.69 tv_ctrlpt.h . . . 556
9.70 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.c File Reference . . . 558
9.71 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.h File Reference . . . 560
9.72 tv_device.h . . . 563
9.73 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionComplete.c File Reference . . . 567
9.73.1 Detailed Description . . . 568
9.73.2 Function Documentation . . . 568
9.73.2.1 UpnpActionComplete_assign() . . . 568
9.73.2.2 UpnpActionComplete_clear_CtrlUrl() . . . 568
9.73.2.3 UpnpActionComplete_delete() . . . 568
9.73.2.4 UpnpActionComplete_dup() . . . 569
9.73.2.5 UpnpActionComplete_get_ActionRequest() . . . 569
9.73.2.6 UpnpActionComplete_get_ActionResult() . . . 569
9.73.2.7 UpnpActionComplete_get_CtrlUrl() . . . 569
9.73.2.8 UpnpActionComplete_get_CtrlUrl_cstr() . . . 569
9.73.2.9 UpnpActionComplete_get_CtrlUrl_Length() . . . 570
9.73.2.10 UpnpActionComplete_get_ErrCode() . . . 570
9.73.2.11 UpnpActionComplete_new() . . . 570
9.73.2.12 UpnpActionComplete_set_ActionRequest() . . . 570
9.73.2.13 UpnpActionComplete_set_ActionResult() . . . 570
9.73.2.14 UpnpActionComplete_set_CtrlUrl() . . . 571
9.73.2.15 UpnpActionComplete_set_ErrCode() . . . 571
9.73.2.16 UpnpActionComplete_strcpy_CtrlUrl() . . . 571
9.73.2.17 UpnpActionComplete_strncpy_CtrlUrl() . . . 571
9.74 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionRequest.c File Reference . . . 572
9.74.1 Detailed Description . . . 573
9.74.2 Function Documentation . . . 573
9.74.2.1 UpnpActionRequest_assign() . . . 573
9.74.2.2 UpnpActionRequest_clear_ActionName() . . . 574
9.74.2.3 UpnpActionRequest_clear_CtrlPtIPAddr() . . . 574
9.74.2.4 UpnpActionRequest_clear_DevUDN() . . . 574
9.74.2.5 UpnpActionRequest_clear_ErrStr() . . . 574
9.74.2.6 UpnpActionRequest_clear_Os() . . . 574
9.74.2.7 UpnpActionRequest_clear_ServiceID() . . . 574
9.74.2.8 UpnpActionRequest_delete() . . . 575
9.74.2.9 UpnpActionRequest_dup() . . . 575
9.74.2.10 UpnpActionRequest_get_ActionName() . . . 575
9.74.2.11 UpnpActionRequest_get_ActionName_cstr() . . . 575
9.74.2.12 UpnpActionRequest_get_ActionName_Length() . . . 575
9.74.2.13 UpnpActionRequest_get_ActionRequest() . . . 576
9.74.2.14 UpnpActionRequest_get_ActionResult() . . . 576
9.74.2.15 UpnpActionRequest_get_CtrlPtIPAddr() . . . 576

9.74.2.16 UpnpActionRequest_get_DevUDN() . . . 576
9.74.2.17 UpnpActionRequest_get_DevUDN_cstr() . . . 576
9.74.2.18 UpnpActionRequest_get_DevUDN_Length() . . . 577
9.74.2.19 UpnpActionRequest_get_ErrCode() . . . 577
9.74.2.20 UpnpActionRequest_get_ErrStr() . . . 577
9.74.2.21 UpnpActionRequest_get_ErrStr_cstr() . . . 577
9.74.2.22 UpnpActionRequest_get_ErrStr_Length() . . . 577
9.74.2.23 UpnpActionRequest_get_Os() . . . 578
9.74.2.24 UpnpActionRequest_get_Os_cstr() . . . 578
9.74.2.25 UpnpActionRequest_get_Os_Length() . . . 578
9.74.2.26 UpnpActionRequest_get_ServiceID() . . . 578
9.74.2.27 UpnpActionRequest_get_ServiceID_cstr() . . . 578
9.74.2.28 UpnpActionRequest_get_ServiceID_Length() . . . 579
9.74.2.29 UpnpActionRequest_get_SoapHeader() . . . 579
9.74.2.30 UpnpActionRequest_get_Socket() . . . 579
9.74.2.31 UpnpActionRequest_new() . . . 579
9.74.2.32 UpnpActionRequest_set_ActionName() . . . 579
9.74.2.33 UpnpActionRequest_set_ActionRequest() . . . 580
9.74.2.34 UpnpActionRequest_set_ActionResult() . . . 580
9.74.2.35 UpnpActionRequest_set_CtrlPtIPAddr() . . . 580
9.74.2.36 UpnpActionRequest_set_DevUDN() . . . 580
9.74.2.37 UpnpActionRequest_set_ErrCode() . . . 580
9.74.2.38 UpnpActionRequest_set_ErrStr() . . . 581
9.74.2.39 UpnpActionRequest_set_Os() . . . 581
9.74.2.40 UpnpActionRequest_set_ServiceID() . . . 581
9.74.2.41 UpnpActionRequest_set_SoapHeader() . . . 581
9.74.2.42 UpnpActionRequest_set_Socket() . . . 581
9.74.2.43 UpnpActionRequest_strcpy_ActionName() . . . 582
9.74.2.44 UpnpActionRequest_strcpy_DevUDN() . . . 582
9.74.2.45 UpnpActionRequest_strcpy_ErrStr() . . . 582
9.74.2.46 UpnpActionRequest_strcpy_Os() . . . 582
9.74.2.47 UpnpActionRequest_strcpy_ServiceID() . . . 583
9.74.2.48 UpnpActionRequest_strncpy_ActionName() . . . 583
9.74.2.49 UpnpActionRequest_strncpy_DevUDN() . . . 583
9.74.2.50 UpnpActionRequest_strncpy_ErrStr() . . . 583
9.74.2.51 UpnpActionRequest_strncpy_Os() . . . 583
9.74.2.52 UpnpActionRequest_strncpy_ServiceID() . . . 584
9.75 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpapi.c File Reference . . . 584
9.76 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpdebug.c File Reference . . . 590
9.76.1 Function Documentation . . . 591
9.76.1.1 UpnpCloseLog() . . . 591
9.76.1.2 UpnpGetDebugFile() . . . 591

9.76.1.3 UpnpInitLog() . . . 592
9.76.1.4 UpnpPrintf() . . . 592
9.76.1.5 UpnpSetLogFileNames() . . . 593
9.76.1.6 UpnpSetLogLevel() . . . 593
9.76.2 Variable Documentation . . . 593
9.76.2.1 g_log_level . . . 593
9.76.2.2 GlobalDebugMutex . . . 594
9.77 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpDiscovery.c File Reference . . . 594
9.77.1 Detailed Description . . . 596
9.77.2 Function Documentation . . . 596
9.77.2.1 UpnpDiscovery_assign() . . . 596
9.77.2.2 UpnpDiscovery_clear_Date() . . . 596
9.77.2.3 UpnpDiscovery_clear_DestAddr() . . . 596
9.77.2.4 UpnpDiscovery_clear_DeviceID() . . . 596
9.77.2.5 UpnpDiscovery_clear_DeviceType() . . . 597
9.77.2.6 UpnpDiscovery_clear_Ext() . . . 597
9.77.2.7 UpnpDiscovery_clear_Location() . . . 597
9.77.2.8 UpnpDiscovery_clear_Os() . . . 597
9.77.2.9 UpnpDiscovery_clear_ServiceType() . . . 597
9.77.2.10 UpnpDiscovery_clear_ServiceVer() . . . 597
9.77.2.11 UpnpDiscovery_delete() . . . 598
9.77.2.12 UpnpDiscovery_dup() . . . 598
9.77.2.13 UpnpDiscovery_get_Date() . . . 598
9.77.2.14 UpnpDiscovery_get_Date_cstr() . . . 598
9.77.2.15 UpnpDiscovery_get_Date_Length() . . . 598
9.77.2.16 UpnpDiscovery_get_DestAddr() . . . 599
9.77.2.17 UpnpDiscovery_get_DeviceID() . . . 599
9.77.2.18 UpnpDiscovery_get_DeviceID_cstr() . . . 599
9.77.2.19 UpnpDiscovery_get_DeviceID_Length() . . . 599
9.77.2.20 UpnpDiscovery_get_DeviceType() . . . 599
9.77.2.21 UpnpDiscovery_get_DeviceType_cstr() . . . 600
9.77.2.22 UpnpDiscovery_get_DeviceType_Length() . . . 600
9.77.2.23 UpnpDiscovery_get_ErrCode() . . . 600
9.77.2.24 UpnpDiscovery_get_Expires() . . . 600
9.77.2.25 UpnpDiscovery_get_Ext() . . . 600
9.77.2.26 UpnpDiscovery_get_Ext_cstr() . . . 600
9.77.2.27 UpnpDiscovery_get_Ext_Length() . . . 601
9.77.2.28 UpnpDiscovery_get_Location() . . . 601
9.77.2.29 UpnpDiscovery_get_Location_cstr() . . . 601
9.77.2.30 UpnpDiscovery_get_Location_Length() . . . 601
9.77.2.31 UpnpDiscovery_get_Os() . . . 601
9.77.2.32 UpnpDiscovery_get_Os_cstr() . . . 602

9.77.2.33 UpnpDiscovery_get_Os_Length() . . . 602
9.77.2.34 UpnpDiscovery_get_ServiceType() . . . 602
9.77.2.35 UpnpDiscovery_get_ServiceType_cstr() . . . 602
9.77.2.36 UpnpDiscovery_get_ServiceType_Length() . . . 602
9.77.2.37 UpnpDiscovery_get_ServiceVer() . . . 603
9.77.2.38 UpnpDiscovery_get_ServiceVer_cstr() . . . 603
9.77.2.39 UpnpDiscovery_get_ServiceVer_Length() . . . 603
9.77.2.40 UpnpDiscovery_new() . . . 603
9.77.2.41 UpnpDiscovery_set_Date() . . . 603
9.77.2.42 UpnpDiscovery_set_DestAddr() . . . 604
9.77.2.43 UpnpDiscovery_set_DeviceID() . . . 604
9.77.2.44 UpnpDiscovery_set_DeviceType() . . . 604
9.77.2.45 UpnpDiscovery_set_ErrCode() . . . 604
9.77.2.46 UpnpDiscovery_set_Expires() . . . 604
9.77.2.47 UpnpDiscovery_set_Ext() . . . 605
9.77.2.48 UpnpDiscovery_set_Location() . . . 605
9.77.2.49 UpnpDiscovery_set_Os() . . . 605
9.77.2.50 UpnpDiscovery_set_ServiceType() . . . 605
9.77.2.51 UpnpDiscovery_set_ServiceVer() . . . 606
9.77.2.52 UpnpDiscovery_strcpy_Date() . . . 606
9.77.2.53 UpnpDiscovery_strcpy_DeviceID() . . . 606
9.77.2.54 UpnpDiscovery_strcpy_DeviceType() . . . 606
9.77.2.55 UpnpDiscovery_strcpy_Ext() . . . 607
9.77.2.56 UpnpDiscovery_strcpy_Location() . . . 607
9.77.2.57 UpnpDiscovery_strcpy_Os() . . . 607
9.77.2.58 UpnpDiscovery_strcpy_ServiceType() . . . 607
9.77.2.59 UpnpDiscovery_strcpy_ServiceVer() . . . 607
9.77.2.60 UpnpDiscovery_strncpy_Date() . . . 608
9.77.2.61 UpnpDiscovery_strncpy_DeviceID() . . . 608
9.77.2.62 UpnpDiscovery_strncpy_DeviceType() . . . 608
9.77.2.63 UpnpDiscovery_strncpy_Ext() . . . 608
9.77.2.64 UpnpDiscovery_strncpy_Location() . . . 609
9.77.2.65 UpnpDiscovery_strncpy_Os() . . . 609
9.77.2.66 UpnpDiscovery_strncpy_ServiceType() . . . 609
9.77.2.67 UpnpDiscovery_strncpy_ServiceVer() . . . 609
9.78 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEvent.c File Reference . . . 610
9.78.1 Detailed Description . . . 611
9.78.2 Function Documentation . . . 611
9.78.2.1 UpnpEvent_assign() . . . 611
9.78.2.2 UpnpEvent_clear_SID() . . . 611
9.78.2.3 UpnpEvent_delete() . . . 611
9.78.2.4 UpnpEvent_dup() . . . 611

9.78.2.5 UpnpEvent_get_ChangedVariables() . . . 612
9.78.2.6 UpnpEvent_get_EventKey() . . . 612
9.78.2.7 UpnpEvent_get_SID() . . . 612
9.78.2.8 UpnpEvent_get_SID_cstr() . . . 612
9.78.2.9 UpnpEvent_get_SID_Length() . . . 612
9.78.2.10 UpnpEvent_new() . . . 613
9.78.2.11 UpnpEvent_set_ChangedVariables() . . . 613
9.78.2.12 UpnpEvent_set_EventKey() . . . 613
9.78.2.13 UpnpEvent_set_SID() . . . 613
9.78.2.14 UpnpEvent_strcpy_SID() . . . 613
9.78.2.15 UpnpEvent_strncpy_SID() . . . 614
9.79 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEventSubscribe.c File Reference . . . 614
9.79.1 Detailed Description . . . 615
9.79.2 Function Documentation . . . 615
9.79.2.1 UpnpEventSubscribe_assign() . . . 615
9.79.2.2 UpnpEventSubscribe_clear_PublisherUrl() . . . 616
9.79.2.3 UpnpEventSubscribe_clear_SID() . . . 616
9.79.2.4 UpnpEventSubscribe_delete() . . . 616
9.79.2.5 UpnpEventSubscribe_dup() . . . 616
9.79.2.6 UpnpEventSubscribe_get_ErrCode() . . . 616
9.79.2.7 UpnpEventSubscribe_get_PublisherUrl() . . . 617
9.79.2.8 UpnpEventSubscribe_get_PublisherUrl_cstr() . . . 617
9.79.2.9 UpnpEventSubscribe_get_PublisherUrl_Length() . . . 617
9.79.2.10 UpnpEventSubscribe_get_SID() . . . 617
9.79.2.11 UpnpEventSubscribe_get_SID_cstr() . . . 617
9.79.2.12 UpnpEventSubscribe_get_SID_Length() . . . 618
9.79.2.13 UpnpEventSubscribe_get_TimeOut() . . . 618
9.79.2.14 UpnpEventSubscribe_new() . . . 618
9.79.2.15 UpnpEventSubscribe_set_ErrCode() . . . 618
9.79.2.16 UpnpEventSubscribe_set_PublisherUrl() . . . 618
9.79.2.17 UpnpEventSubscribe_set_SID() . . . 619
9.79.2.18 UpnpEventSubscribe_set_TimeOut() . . . 619
9.79.2.19 UpnpEventSubscribe_strcpy_PublisherUrl() . . . 619
9.79.2.20 UpnpEventSubscribe_strcpy_SID() . . . 619
9.79.2.21 UpnpEventSubscribe_strncpy_PublisherUrl() . . . 620
9.79.2.22 UpnpEventSubscribe_strncpy_SID() . . . 620
9.80 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpExtraHeaders.c File Reference . . . 620
9.80.1 Detailed Description . . . 621
9.80.2 Function Documentation . . . 621
9.80.2.1 UpnpExtraHeaders_add_to_list_node() . . . 621
9.80.2.2 UpnpExtraHeaders_assign() . . . 622
9.80.2.3 UpnpExtraHeaders_clear_name() . . . 622

9.80.2.4 UpnpExtraHeaders_clear_value() . . . 622
9.80.2.5 UpnpExtraHeaders_delete() . . . 622
9.80.2.6 UpnpExtraHeaders_dup() . . . 622
9.80.2.7 UpnpExtraHeaders_get_name() . . . 623
9.80.2.8 UpnpExtraHeaders_get_name_cstr() . . . 623
9.80.2.9 UpnpExtraHeaders_get_name_Length() . . . 623
9.80.2.10 UpnpExtraHeaders_get_node() . . . 623
9.80.2.11 UpnpExtraHeaders_get_resp() . . . 623
9.80.2.12 UpnpExtraHeaders_get_resp_cstr() . . . 623
9.80.2.13 UpnpExtraHeaders_get_value() . . . 624
9.80.2.14 UpnpExtraHeaders_get_value_cstr() . . . 624
9.80.2.15 UpnpExtraHeaders_get_value_Length() . . . 624
9.80.2.16 UpnpExtraHeaders_new() . . . 624
9.80.2.17 UpnpExtraHeaders_set_name() . . . 624
9.80.2.18 UpnpExtraHeaders_set_node() . . . 625
9.80.2.19 UpnpExtraHeaders_set_resp() . . . 625
9.80.2.20 UpnpExtraHeaders_set_value() . . . 625
9.80.2.21 UpnpExtraHeaders_strcpy_name() . . . 625
9.80.2.22 UpnpExtraHeaders_strcpy_value() . . . 625
9.80.2.23 UpnpExtraHeaders_strncpy_name() . . . 626
9.80.2.24 UpnpExtraHeaders_strncpy_value() . . . 626
9.81 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpFileInfo.c File Reference . . . 626
9.81.1 Detailed Description . . . 627
9.81.2 Function Documentation . . . 627
9.81.2.1 UpnpFileInfo_add_to_list_ExtraHeadersList() . . . 627
9.81.2.2 UpnpFileInfo_assign() . . . 628
9.81.2.3 UpnpFileInfo_clear_CtrlPtIPAddr() . . . 628
9.81.2.4 UpnpFileInfo_clear_Os() . . . 628
9.81.2.5 UpnpFileInfo_delete() . . . 628
9.81.2.6 UpnpFileInfo_dup() . . . 628
9.81.2.7 UpnpFileInfo_get_ContentType() . . . 629
9.81.2.8 UpnpFileInfo_get_ContentType_cstr() . . . 629
9.81.2.9 UpnpFileInfo_get_CtrlPtIPAddr() . . . 629
9.81.2.10 UpnpFileInfo_get_ExtraHeadersList() . . . 629
9.81.2.11 UpnpFileInfo_get_FileLength() . . . 629
9.81.2.12 UpnpFileInfo_get_IsDirectory() . . . 629
9.81.2.13 UpnpFileInfo_get_IsReadable() . . . 630
9.81.2.14 UpnpFileInfo_get_LastModified() . . . 630
9.81.2.15 UpnpFileInfo_get_Os() . . . 630
9.81.2.16 UpnpFileInfo_get_Os_cstr() . . . 630
9.81.2.17 UpnpFileInfo_get_Os_Length() . . . 630
9.81.2.18 UpnpFileInfo_new() . . . 631

9.81.2.19 UpnpFileInfo_set_ContentType() . . . 631
9.81.2.20 UpnpFileInfo_set_CtrlPtIPAddr() . . . 631
9.81.2.21 UpnpFileInfo_set_ExtraHeadersList() . . . 631
9.81.2.22 UpnpFileInfo_set_FileLength() . . . 631
9.81.2.23 UpnpFileInfo_set_IsDirectory() . . . 632
9.81.2.24 UpnpFileInfo_set_IsReadable() . . . 632
9.81.2.25 UpnpFileInfo_set_LastModified() . . . 632
9.81.2.26 UpnpFileInfo_set_Os() . . . 632
9.81.2.27 UpnpFileInfo_strcpy_Os() . . . 632
9.81.2.28 UpnpFileInfo_strncpy_Os() . . . 633
9.82 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarComplete.c File Reference . . . 633
9.82.1 Detailed Description . . . 634
9.82.2 Function Documentation . . . 634
9.82.2.1 UpnpStateVarComplete_assign() . . . 634
9.82.2.2 UpnpStateVarComplete_clear_CtrlUrl() . . . 635
9.82.2.3 UpnpStateVarComplete_clear_StateVarName() . . . 635
9.82.2.4 UpnpStateVarComplete_delete() . . . 635
9.82.2.5 UpnpStateVarComplete_dup() . . . 635
9.82.2.6 UpnpStateVarComplete_get_CtrlUrl() . . . 635
9.82.2.7 UpnpStateVarComplete_get_CtrlUrl_cstr() . . . 636
9.82.2.8 UpnpStateVarComplete_get_CtrlUrl_Length() . . . 636
9.82.2.9 UpnpStateVarComplete_get_CurrentVal() . . . 636
9.82.2.10 UpnpStateVarComplete_get_CurrentVal_cstr() . . . 636
9.82.2.11 UpnpStateVarComplete_get_ErrCode() . . . 636
9.82.2.12 UpnpStateVarComplete_get_StateVarName() . . . 637
9.82.2.13 UpnpStateVarComplete_get_StateVarName_cstr() . . . 637
9.82.2.14 UpnpStateVarComplete_get_StateVarName_Length() . . . 637
9.82.2.15 UpnpStateVarComplete_new() . . . 637
9.82.2.16 UpnpStateVarComplete_set_CtrlUrl() . . . 637
9.82.2.17 UpnpStateVarComplete_set_CurrentVal() . . . 638
9.82.2.18 UpnpStateVarComplete_set_ErrCode() . . . 638
9.82.2.19 UpnpStateVarComplete_set_StateVarName() . . . 638
9.82.2.20 UpnpStateVarComplete_strcpy_CtrlUrl() . . . 638
9.82.2.21 UpnpStateVarComplete_strcpy_StateVarName() . . . 639
9.82.2.22 UpnpStateVarComplete_strncpy_CtrlUrl() . . . 639
9.82.2.23 UpnpStateVarComplete_strncpy_StateVarName() . . . 639
9.83 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarRequest.c File Reference . . . 639
9.83.1 Detailed Description . . . 641
9.83.2 Function Documentation . . . 641
9.83.2.1 UpnpStateVarRequest_assign() . . . 641
9.83.2.2 UpnpStateVarRequest_clear_CtrlPtIPAddr() . . . 641
9.83.2.3 UpnpStateVarRequest_clear_DevUDN() . . . 641

9.83.2.4 UpnpStateVarRequest_clear_ErrStr() . . . 641
9.83.2.5 UpnpStateVarRequest_clear_ServiceID() . . . 642
9.83.2.6 UpnpStateVarRequest_clear_StateVarName() . . . 642
9.83.2.7 UpnpStateVarRequest_delete() . . . 642
9.83.2.8 UpnpStateVarRequest_dup() . . . 642
9.83.2.9 UpnpStateVarRequest_get_CtrlPtIPAddr() . . . 642
9.83.2.10 UpnpStateVarRequest_get_CurrentVal() . . . 643
9.83.2.11 UpnpStateVarRequest_get_CurrentVal_cstr() . . . 643
9.83.2.12 UpnpStateVarRequest_get_DevUDN() . . . 643
9.83.2.13 UpnpStateVarRequest_get_DevUDN_cstr() . . . 643
9.83.2.14 UpnpStateVarRequest_get_DevUDN_Length() . . . 643
9.83.2.15 UpnpStateVarRequest_get_ErrCode() . . . 644
9.83.2.16 UpnpStateVarRequest_get_ErrStr() . . . 644
9.83.2.17 UpnpStateVarRequest_get_ErrStr_cstr() . . . 644
9.83.2.18 UpnpStateVarRequest_get_ErrStr_Length() . . . 644
9.83.2.19 UpnpStateVarRequest_get_ServiceID() . . . 644
9.83.2.20 UpnpStateVarRequest_get_ServiceID_cstr() . . . 645
9.83.2.21 UpnpStateVarRequest_get_ServiceID_Length() . . . 645
9.83.2.22 UpnpStateVarRequest_get_Socket() . . . 645
9.83.2.23 UpnpStateVarRequest_get_StateVarName() . . . 645
9.83.2.24 UpnpStateVarRequest_get_StateVarName_cstr() . . . 645
9.83.2.25 UpnpStateVarRequest_get_StateVarName_Length() . . . 646
9.83.2.26 UpnpStateVarRequest_new() . . . 646
9.83.2.27 UpnpStateVarRequest_set_CtrlPtIPAddr() . . . 646
9.83.2.28 UpnpStateVarRequest_set_CurrentVal() . . . 646
9.83.2.29 UpnpStateVarRequest_set_DevUDN() . . . 646
9.83.2.30 UpnpStateVarRequest_set_ErrCode() . . . 647
9.83.2.31 UpnpStateVarRequest_set_ErrStr() . . . 647
9.83.2.32 UpnpStateVarRequest_set_ServiceID() . . . 647
9.83.2.33 UpnpStateVarRequest_set_Socket() . . . 647
9.83.2.34 UpnpStateVarRequest_set_StateVarName() . . . 647
9.83.2.35 UpnpStateVarRequest_strcpy_DevUDN() . . . 648
9.83.2.36 UpnpStateVarRequest_strcpy_ErrStr() . . . 648
9.83.2.37 UpnpStateVarRequest_strcpy_ServiceID() . . . 648
9.83.2.38 UpnpStateVarRequest_strcpy_StateVarName() . . . 648
9.83.2.39 UpnpStateVarRequest_strncpy_DevUDN() . . . 649
9.83.2.40 UpnpStateVarRequest_strncpy_ErrStr() . . . 649
9.83.2.41 UpnpStateVarRequest_strncpy_ServiceID() . . . 649
9.83.2.42 UpnpStateVarRequest_strncpy_StateVarName() . . . 649
9.84 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpString.c File Reference . . . 650
9.84.1 Detailed Description . . . 651
9.85 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpSubscriptionRequest.c File Reference . . . 651

9.85.1 Detailed Description . . . 652
9.85.2 Function Documentation . . . 652
9.85.2.1 UpnpSubscriptionRequest_assign() . . . 652
9.85.2.2 UpnpSubscriptionRequest_clear_ServiceId() . . . 653
9.85.2.3 UpnpSubscriptionRequest_clear_SID() . . . 653
9.85.2.4 UpnpSubscriptionRequest_clear_UDN() . . . 653
9.85.2.5 UpnpSubscriptionRequest_delete() . . . 653
9.85.2.6 UpnpSubscriptionRequest_dup() . . . 653
9.85.2.7 UpnpSubscriptionRequest_get_ServiceId() . . . 654
9.85.2.8 UpnpSubscriptionRequest_get_ServiceId_cstr() . . . 654
9.85.2.9 UpnpSubscriptionRequest_get_ServiceId_Length() . . . 654
9.85.2.10 UpnpSubscriptionRequest_get_SID() . . . 654
9.85.2.11 UpnpSubscriptionRequest_get_SID_cstr() . . . 654
9.85.2.12 UpnpSubscriptionRequest_get_SID_Length() . . . 655
9.85.2.13 UpnpSubscriptionRequest_get_UDN() . . . 655
9.85.2.14 UpnpSubscriptionRequest_get_UDN_cstr() . . . 655
9.85.2.15 UpnpSubscriptionRequest_get_UDN_Length() . . . 655
9.85.2.16 UpnpSubscriptionRequest_new() . . . 655
9.85.2.17 UpnpSubscriptionRequest_set_ServiceId() . . . 656
9.85.2.18 UpnpSubscriptionRequest_set_SID() . . . 656
9.85.2.19 UpnpSubscriptionRequest_set_UDN() . . . 656
9.85.2.20 UpnpSubscriptionRequest_strcpy_ServiceId() . . . 656
9.85.2.21 UpnpSubscriptionRequest_strcpy_SID() . . . 657
9.85.2.22 UpnpSubscriptionRequest_strcpy_UDN() . . . 657
9.85.2.23 UpnpSubscriptionRequest_strncpy_ServiceId() . . . 657
9.85.2.24 UpnpSubscriptionRequest_strncpy_SID() . . . 657
9.85.2.25 UpnpSubscriptionRequest_strncpy_UDN() . . . 657
9.86 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnptools.c File Reference . . . 658
9.86.1 Macro Definition Documentation . . . 659
9.86.1.1 HEADER_LENGTH . . . 659
9.86.2 Function Documentation . . . 659
9.86.2.1 addToAction() . . . 659
9.86.2.2 makeAction() . . . 660
9.86.3 Variable Documentation . . . 660
9.86.3.1 ErrorMessages . . . 660
9.87 /build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_ctrlpt.c File Reference . . . 661
9.87.1 Function Documentation . . . 662
9.87.1.1 free_subscribe_arg() . . . 662
9.87.1.2 gena_process_notification_event() . . . 662
9.87.1.3 gena_subscribe() . . . 663
9.87.1.4 gena_unsubscribe() . . . 663
9.87.1.5 GenaAutoRenewSubscription() . . . 664

9.87.1.6 genaRenewSubscription() . . . 664
9.87.1.7 genaSubscribe() . . . 665
9.87.1.8 genaUnregisterClient() . . . 665
9.87.1.9 genaUnSubscribe() . . . 666
9.87.1.10 ScheduleGenaAutoRenew() . . . 666
9.88 /build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_device.c File Reference . . . 667
9.88.1 Function Documentation . . . 668
9.88.1.1 AllocGenaHeaders() . . . 668
9.88.1.2 create_url_list() . . . 669
9.88.1.3 free_notify_struct() . . . 669
9.88.1.4 gena_process_subscription_renewal_request() . . . 670
9.88.1.5 gena_process_subscription_request() . . . 670
9.88.1.6 gena_process_unsubscribe_request() . . . 670
9.88.1.7 gena_validate_delivery_urls() . . . 671
9.88.1.8 genaInitNotify() . . . 671
9.88.1.9 genaInitNotifyExt() . . . 672
9.88.1.10 genaNotify() . . . 672
9.88.1.11 genaNotifyAll() . . . 673
9.88.1.12 genaNotifyAllExt() . . . 674
9.88.1.13 genaNotifyThread() . . . 674
9.88.1.14 genaUnregisterDevice() . . . 675
9.88.1.15 GeneratePropertySet() . . . 675
9.88.1.16 notify_send_and_recv() . . . 676
9.88.1.17 respond_ok() . . . 676
9.89 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/client_table/GenlibClientSubscription.c File Reference . . . 677
9.89.1 Detailed Description . . . 678
9.89.2 Function Documentation . . . 678
9.89.2.1 GenlibClientSubscription_assign() . . . 678
9.89.2.2 GenlibClientSubscription_clear_ActualSID() . . . 679
9.89.2.3 GenlibClientSubscription_clear_EventURL() . . . 679
9.89.2.4 GenlibClientSubscription_clear_SID() . . . 679
9.89.2.5 GenlibClientSubscription_delete() . . . 679
9.89.2.6 GenlibClientSubscription_dup() . . . 679
9.89.2.7 GenlibClientSubscription_get_ActualSID() . . . 680
9.89.2.8 GenlibClientSubscription_get_ActualSID_cstr() . . . 680
9.89.2.9 GenlibClientSubscription_get_ActualSID_Length() . . . 680
9.89.2.10 GenlibClientSubscription_get_EventURL() . . . 680
9.89.2.11 GenlibClientSubscription_get_EventURL_cstr() . . . 680
9.89.2.12 GenlibClientSubscription_get_EventURL_Length() . . . 681
9.89.2.13 GenlibClientSubscription_get_Next() . . . 681
9.89.2.14 GenlibClientSubscription_get_RenewEventId() . . . 681

9.89.2.15 GenlibClientSubscription_get_SID() . . . 681
9.89.2.16 GenlibClientSubscription_get_SID_cstr() . . . 681
9.89.2.17 GenlibClientSubscription_get_SID_Length() . . . 682
9.89.2.18 GenlibClientSubscription_new() . . . 682
9.89.2.19 GenlibClientSubscription_set_ActualSID() . . . 682
9.89.2.20 GenlibClientSubscription_set_EventURL() . . . 682
9.89.2.21 GenlibClientSubscription_set_Next() . . . 682
9.89.2.22 GenlibClientSubscription_set_RenewEventId() . . . 683
9.89.2.23 GenlibClientSubscription_set_SID() . . . 683
9.89.2.24 GenlibClientSubscription_strcpy_ActualSID() . . . 683
9.89.2.25 GenlibClientSubscription_strcpy_EventURL() . . . 683
9.89.2.26 GenlibClientSubscription_strcpy_SID() . . . 684
9.89.2.27 GenlibClientSubscription_strncpy_ActualSID() . . . 684
9.89.2.28 GenlibClientSubscription_strncpy_EventURL() . . . 684
9.89.2.29 GenlibClientSubscription_strncpy_SID() . . . 684
9.90 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/miniserver/miniserver.c File Reference . . . 685
9.90.1 Detailed Description . . . 686
9.90.2 Macro Definition Documentation . . . 686
9.90.2.1 APPLICATION_LISTENING_PORT . . . 686
9.90.3 Enumeration Type Documentation . . . 686
9.90.3.1 MiniServerState . . . 686
9.90.4 Function Documentation . . . 687
9.90.4.1 get_miniserver_stopsock() . . . 687
9.90.4.2 get_port() . . . 687
9.90.4.3 RunMiniServer() . . . 688
9.90.4.4 StartMiniServer() . . . 688
9.90.4.5 StopMiniServer() . . . 689
9.90.5 Variable Documentation . . . 689
9.90.5.1 gMServState . . . 689
9.90.5.2 miniStopSockPort . . . 689
9.91 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/httpreadwrite.c File Reference . . . 690
9.91.1 Detailed Description . . . 692
9.91.2 Function Documentation . . . 692
9.91.2.1 Check_Connect_And_Wait_Connection() . . . 692
9.91.2.2 http_CloseHttpConnection() . . . 692
9.91.2.3 http_Connect() . . . 693
9.91.2.4 http_EndHttpRequest() . . . 693
9.91.2.5 http_FixStrUrl() . . . 694
9.91.2.6 http_FixUrl() . . . 694
9.91.2.7 http_GetHttpResponse() . . . 695
9.91.2.8 http_MakeHttpRequest() . . . 696
9.91.2.9 http_MakeMessage() . . . 697

9.91.2.10 http_OpenHttpConnection() . . . 698
9.91.2.11 http_ReadHttpResponse() . . . 699
9.91.2.12 http_RecvMessage() . . . 699
9.91.2.13 http_SendMessage() . . . 700
9.91.2.14 http_WriteHttpRequest() . . . 701
9.91.2.15 ReadResponseLineAndHeaders() . . . 701
9.92 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/parsetools.c File Reference . . . 702
9.92.1 Detailed Description . . . 702
9.92.2 Function Documentation . . . 703
9.92.2.1 has_xml_content_type() . . . 703
9.93 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/webserver.c File Reference . . . 703
9.93.1 Detailed Description . . . 706
9.93.2 Enumeration Type Documentation . . . 706
9.93.2.1 resp_type . . . 706
9.93.3 Function Documentation . . . 706
9.93.3.1 alias_grab() . . . 706
9.93.3.2 alias_release() . . . 706
9.93.3.3 CheckOtherHTTPHeaders() . . . 707
9.93.3.4 CreateHTTPRangeResponseHeader() . . . 707
9.93.3.5 ExtraHTTPHeaders() . . . 708
9.93.3.6 FreeExtraHTTPHeaders() . . . 708
9.93.3.7 get_alias() . . . 708
9.93.3.8 get_content_type() . . . 709
9.93.3.9 get_file_info() . . . 709
9.93.3.10 GetNextRange() . . . 710
9.93.3.11 glob_alias_init() . . . 710
9.93.3.12 http_RecvPostMessage() . . . 711
9.93.3.13 is_valid_alias() . . . 711
9.93.3.14 isFileInVirtualDir() . . . 712
9.93.3.15 media_list_init() . . . 712
9.93.3.16 process_request() . . . 712
9.93.3.17 search_extension() . . . 713
9.93.3.18 StrStr() . . . 714
9.93.3.19 StrTok() . . . 714
9.93.3.20 ToUpperCase() . . . 714
9.93.3.21 web_server_callback() . . . 715
9.93.3.22 web_server_destroy() . . . 715
9.93.3.23 web_server_init() . . . 715
9.93.3.24 web_server_set_alias() . . . 716
9.93.3.25 web_server_set_cors() . . . 716
9.93.3.26 web_server_set_root_dir() . . . 717
9.93.4 Variable Documentation . . . 717

9.93.4.1 gAliasDoc . . . 717
9.93.4.2 gMediaTypeList . . . 717
9.93.4.3 gMediaTypes . . . 718
9.94 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/sock.c File Reference . . . 718
9.94.1 Detailed Description . . . 719
9.95 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/uri/uri.c File Reference . . . 719
9.95.1 Detailed Description . . . 720
9.95.2 Function Documentation . . . 721
9.95.2.1 copy_token() . . . 721
9.95.2.2 copy_URL_list() . . . 721
9.95.2.3 free_URL_list() . . . 721
9.95.2.4 is_escaped() . . . 722
9.95.2.5 is_mark() . . . 722
9.95.2.6 is_reserved() . . . 723
9.95.2.7 is_unreserved() . . . 723
9.95.2.8 parse_hostport() . . . 723
9.95.2.9 parse_scheme() . . . 725
9.95.2.10 parse_uri() . . . 725
9.95.2.11 parse_uric() . . . 726
9.95.2.12 remove_dots() . . . 726
9.95.2.13 remove_escaped_chars() . . . 727
9.95.2.14 replace_escaped() . . . 727
9.95.2.15 resolve_rel_url() . . . 728
9.95.2.16 token_cmp() . . . 729
9.95.2.17 token_string_casecmp() . . . 729
9.96 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/util/util.c File Reference . . . 729
9.96.1 Detailed Description . . . 730
9.96.2 Function Documentation . . . 730
9.96.2.1 linecopy() . . . 730
9.96.2.2 linecopylen() . . . 730
9.96.2.3 namecopy() . . . 731
9.97 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/client_table.h File Reference . . . 731
9.97.1 Function Documentation . . . 732
9.97.1.1 free_client_subscription() . . . 732
9.97.1.2 freeClientSubList() . . . 732
9.97.1.3 GetClientSubActualSID() . . . 733
9.97.1.4 GetClientSubClientSID() . . . 733
9.97.1.5 RemoveClientSubClientSID() . . . 734
9.98 client_table.h . . . 734
9.99 config.h . . . 735
9.100 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena.h File Reference . . . 737
9.100.1 Macro Definition Documentation . . . 739

9.100.1.1 SubscribeLock . . . 739
9.100.1.2 SubscribeUnlock . . . 740
9.100.1.3 XML_VERSION . . . 740
9.100.2 Typedef Documentation . . . 740
9.100.2.1 notify_thread_struct . . . 740
9.100.3 Function Documentation . . . 740
9.100.3.1 error_respond() . . . 740
9.100.3.2 genaCallback() . . . 741
9.100.3.3 genaInitNotify() . . . 741
9.100.3.4 genaInitNotifyExt() . . . 742
9.100.3.5 genaNotifyAll() . . . 742
9.100.3.6 genaNotifyAllExt() . . . 743
9.100.3.7 genaRenewSubscription() . . . 743
9.100.3.8 genaSubscribe() . . . 744
9.100.3.9 genaUnregisterClient() . . . 745
9.100.3.10 genaUnregisterDevice() . . . 745
9.100.3.11 genaUnSubscribe() . . . 746
9.101 gena.h . . . 746
9.102 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_ctrlpt.h File Reference . . . 749
9.102.1 Function Documentation . . . 749
9.102.1.1 gena_process_notification_event() . . . 749
9.103 gena_ctrlpt.h . . . 750
9.104 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_device.h File Reference . . . 750
9.104.1 Function Documentation . . . 751
9.104.1.1 gena_process_subscription_renewal_request() . . . 751
9.104.1.2 gena_process_subscription_request() . . . 752
9.104.1.3 gena_process_unsubscribe_request() . . . 752
9.105 gena_device.h . . . 753
9.106 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/GenlibClientSubscription.h File Reference . . . 753
9.106.1 Detailed Description . . . 755
9.106.2 Typedef Documentation . . . 755
9.106.2.1 GenlibClientSubscription . . . 755
9.106.3 Function Documentation . . . 756
9.106.3.1 GenlibClientSubscription_assign() . . . 756
9.106.3.2 GenlibClientSubscription_clear_ActualSID() . . . 756
9.106.3.3 GenlibClientSubscription_clear_EventURL() . . . 756
9.106.3.4 GenlibClientSubscription_clear_SID() . . . 756
9.106.3.5 GenlibClientSubscription_delete() . . . 756
9.106.3.6 GenlibClientSubscription_dup() . . . 757
9.106.3.7 GenlibClientSubscription_get_ActualSID() . . . 757
9.106.3.8 GenlibClientSubscription_get_ActualSID_cstr() . . . 757
9.106.3.9 GenlibClientSubscription_get_ActualSID_Length() . . . 757

9.106.3.10 GenlibClientSubscription_get_EventURL() . . . 757
9.106.3.11 GenlibClientSubscription_get_EventURL_cstr() . . . 758
9.106.3.12 GenlibClientSubscription_get_EventURL_Length() . . . 758
9.106.3.13 GenlibClientSubscription_get_Next() . . . 758
9.106.3.14 GenlibClientSubscription_get_RenewEventId() . . . 758
9.106.3.15 GenlibClientSubscription_get_SID() . . . 758
9.106.3.16 GenlibClientSubscription_get_SID_cstr() . . . 759
9.106.3.17 GenlibClientSubscription_get_SID_Length() . . . 759
9.106.3.18 GenlibClientSubscription_new() . . . 759
9.106.3.19 GenlibClientSubscription_set_ActualSID() . . . 759
9.106.3.20 GenlibClientSubscription_set_EventURL() . . . 759
9.106.3.21 GenlibClientSubscription_set_Next() . . . 760
9.106.3.22 GenlibClientSubscription_set_RenewEventId() . . . 760
9.106.3.23 GenlibClientSubscription_set_SID() . . . 760
9.106.3.24 GenlibClientSubscription_strcpy_ActualSID() . . . 760
9.106.3.25 GenlibClientSubscription_strcpy_EventURL() . . . 760
9.106.3.26 GenlibClientSubscription_strcpy_SID() . . . 761
9.106.3.27 GenlibClientSubscription_strncpy_ActualSID() . . . 761
9.106.3.28 GenlibClientSubscription_strncpy_EventURL() . . . 761
9.106.3.29 GenlibClientSubscription_strncpy_SID() . . . 761
9.107 GenlibClientSubscription.h . . . 762
9.108 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h File Reference . . . 763
9.108.1 Enumeration Type Documentation . . . 765
9.108.1.1 parse_status_t . . . 765
9.108.2 Function Documentation . . . 765
9.108.2.1 print_http_headers() . . . 765
9.109 httpparser.h . . . 766
9.110 httpreadwrite.h . . . 771
9.111 inet_pton.h . . . 775
9.112 md5.h . . . 775
9.113 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/membuffer.h File Reference . . . 776
9.113.1 Macro Definition Documentation . . . 777
9.113.1.1 MEMBUF_DEF_SIZE_INC . . . 777
9.113.2 Function Documentation . . . 777
9.113.2.1 membuffer_append() . . . 777
9.113.2.2 membuffer_append_str() . . . 778
9.113.2.3 membuffer_assign() . . . 778
9.113.2.4 membuffer_assign_str() . . . 779
9.113.2.5 membuffer_attach() . . . 779
9.113.2.6 membuffer_delete() . . . 779
9.113.2.7 membuffer_destroy() . . . 780
9.113.2.8 membuffer_detach() . . . 780

9.113.2.9 membuffer_init() . . . 780
9.113.2.10 membuffer_insert() . . . 780
9.113.2.11 membuffer_set_size() . . . 781
9.113.2.12 memptr_cmp() . . . 781
9.113.2.13 memptr_cmp_nocase() . . . 782
9.113.2.14 str_alloc() . . . 782
9.114 membuffer.h . . . 783
9.115 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/miniserver.h File Reference . . . 784
9.115.1 Typedef Documentation . . . 786
9.115.1.1 MiniServerCallback . . . 786
9.115.2 Function Documentation . . . 786
9.115.2.1 SetGenaCallback() . . . 786
9.115.2.2 SetHTTPGetCallback() . . . 786
9.115.2.3 SetSoapCallback() . . . 786
9.115.2.4 StartMiniServer() . . . 787
9.115.2.5 StopMiniServer() . . . 787
9.116 miniserver.h . . . 788
9.117 netall.h . . . 789
9.118 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/parsetools.h File Reference . . . 789
9.118.1 Function Documentation . . . 790
9.118.1.1 has_xml_content_type() . . . 790
9.119 parsetools.h . . . 790
9.120 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/service_table.h File Reference . . . 791
9.120.1 Function Documentation . . . 793
9.120.1.1 addServiceTable() . . . 793
9.120.1.2 copy_subscription() . . . 793
9.120.1.3 FindServiceControlURLPath() . . . 794
9.120.1.4 FindServiceEventURLPath() . . . 794
9.120.1.5 FindServiceId() . . . 794
9.120.1.6 freeService() . . . 795
9.120.1.7 freeServiceList() . . . 795
9.120.1.8 freeServiceTable() . . . 795
9.120.1.9 freeSubscription() . . . 796
9.120.1.10 freeSubscriptionList() . . . 796
9.120.1.11 getElementValue() . . . 796
9.120.1.12 GetFirstSubscription() . . . 797
9.120.1.13 GetNextSubscription() . . . 797
9.120.1.14 getServiceTable() . . . 797
9.120.1.15 getSubElement() . . . 798
9.120.1.16 GetSubscriptionSID() . . . 798
9.120.1.17 printService() . . . 798
9.120.1.18 printServiceList() . . . 799

9.120.1.19 printServiceTable() . . . 799
9.120.1.20 removeServiceTable() . . . 800
9.120.1.21 RemoveSubscriptionSID() . . . 800
9.121 service_table.h . . . 800
9.122 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/soaplib.h File Reference . . . 803
9.122.1 Function Documentation . . . 804
9.122.1.1 soap_device_callback() . . . 804
9.123 soaplib.h . . . 804
9.124 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sock.h File Reference . . . 806
9.125 sock.h . . . 807
9.126 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h File Reference . . . 809
9.127 ssdplib.h . . . 811
9.128 statcodes.h . . . 816
9.129 strintmap.h . . . 817
9.130 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sysdep.h File Reference . . . 818
9.130.1 Macro Definition Documentation . . . 820
9.130.1.1 UUIDS_PER_TICK . . . 820
9.130.2 Function Documentation . . . 820
9.130.2.1 get_ieee_node_identifier() . . . 820
9.130.2.2 get_system_time() . . . 820
9.131 sysdep.h . . . 821
9.132 unixutil.h . . . 821
9.133 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnpapi.h File Reference . . . 822
9.133.1 Macro Definition Documentation . . . 824
9.133.1.1 HandleReadLock . . . 824
9.133.1.2 HandleUnlock . . . 824
9.133.1.3 HandleWriteLock . . . 825
9.133.2 Variable Documentation . . . 825
9.133.2.1 gUpnpSdkNLSuuid . . . 825
9.134 upnpapi.h . . . 825
9.135 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnputil.h File Reference . . . 828
9.135.1 Function Documentation . . . 829
9.135.1.1 linecopy() . . . 829
9.135.1.2 linecopylen() . . . 829
9.135.1.3 namecopy() . . . 829
9.136 upnputil.h . . . 830
9.137 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h File Reference . . . 831
9.137.1 Macro Definition Documentation . . . 833
9.137.1.1 RESERVED . . . 833
9.137.2 Typedef Documentation . . . 833
9.137.2.1 hostport_type . . . 833
9.137.2.2 token . . . 833

9.137.2.3 uri_type . . . 833
9.137.2.4 URL_list . . . 833
9.137.3 Function Documentation . . . 833
9.137.3.1 copy_URL_list() . . . 833
9.137.3.2 free_URL_list() . . . 834
9.137.3.3 parse_token() . . . 834
9.137.3.4 parse_uri() . . . 834
9.137.3.5 print_token() . . . 835
9.137.3.6 print_uri() . . . 835
9.137.3.7 remove_dots() . . . 836
9.137.3.8 remove_escaped_chars() . . . 836
9.137.3.9 replace_escaped() . . . 837
9.137.3.10 resolve_rel_url() . . . 837
9.137.3.11 token_cmp() . . . 838
9.137.3.12 token_string_casecmp() . . . 838
9.138 uri.h . . . 839
9.139 urlconfig.h . . . 841
9.140 uuid.h . . . 842
9.141 VirtualDir.h . . . 843
9.142 webserver.h . . . 843
9.143 /build/reproducible-path/pupnp-1.14.20/upnp/src/inet_pton.c File Reference . . . 844
9.144 /build/reproducible-path/pupnp-1.14.20/upnp/src/soap/soap_device.c File Reference . . . 844
9.144.1 Macro Definition Documentation . . . 846
9.144.1.1 SOAP_TIMEOUT . . . 846
9.144.2 Function Documentation . . . 846
9.144.2.1 check_soap_request() . . . 846
9.144.2.2 check_soapaction_hdr() . . . 847
9.144.2.3 get_dev_service() . . . 847
9.144.2.4 get_mpost_acton_hdrval() . . . 848
9.144.2.5 handle_invoke_action() . . . 848
9.144.2.6 handle_query_variable() . . . 849
9.144.2.7 send_action_response() . . . 849
9.144.2.8 send_error_response() . . . 850
9.144.2.9 send_var_query_response() . . . 850
9.144.2.10 soap_device_callback() . . . 850
9.144.3 Variable Documentation . . . 851
9.144.3.1 SOAP_URN . . . 851
9.145 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_ctrlpt.c File Reference . . . 851
9.146 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_device.c File Reference . . . 852
9.147 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_server.c File Reference . . . 854
9.148 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.c File Reference . . . 855
9.148.1 Detailed Description . . . 856

9.148.2 Function Documentation . . . 857
9.148.2.1 SSDPResultData_assign() . . . 857
9.148.2.2 SSDPResultData_delete() . . . 857
9.148.2.3 SSDPResultData_dup() . . . 857
9.148.2.4 SSDPResultData_get_Cookie() . . . 857
9.148.2.5 SSDPResultData_get_CtrlptCallback() . . . 857
9.148.2.6 SSDPResultData_get_Param() . . . 858
9.148.2.7 SSDPResultData_new() . . . 858
9.148.2.8 SSDPResultData_set_Cookie() . . . 858
9.148.2.9 SSDPResultData_set_CtrlptCallback() . . . 858
9.148.2.10 SSDPResultData_set_Param() . . . 858
9.149 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.h File Reference . . . 859
9.149.1 Detailed Description . . . 860
9.149.2 Typedef Documentation . . . 860
9.149.2.1 SSDPResultData . . . 860
9.149.3 Function Documentation . . . 860
9.149.3.1 SSDPResultData_assign() . . . 860
9.149.3.2 SSDPResultData_delete() . . . 861
9.149.3.3 SSDPResultData_dup() . . . 861
9.149.3.4 SSDPResultData_get_Cookie() . . . 861
9.149.3.5 SSDPResultData_get_CtrlptCallback() . . . 861
9.149.3.6 SSDPResultData_get_Param() . . . 861
9.149.3.7 SSDPResultData_new() . . . 862
9.149.3.8 SSDPResultData_set_Cookie() . . . 862
9.149.3.9 SSDPResultData_set_CtrlptCallback() . . . 862
9.149.3.10 SSDPResultData_set_Param() . . . 862
9.150 SSDPResultData.h . . . 863
9.151 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultDataCallback.c File Reference . . . 863
9.151.1 Detailed Description . . . 864
9.152 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultDataCallback.h File Reference . . . 864
9.152.1 Detailed Description . . . 865
9.153 SSDPResultDataCallback.h . . . 866
9.154 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/FreeList.h File Reference . . . 866
9.154.1 Typedef Documentation . . . 867
9.154.1.1 FreeList . . . 867
9.154.1.2 FreeListNode . . . 867
9.154.2 Function Documentation . . . 867
9.154.2.1 FreeListAlloc() . . . 867
9.154.2.2 FreeListDestroy() . . . 868
9.154.2.3 FreeListFree() . . . 868
9.154.2.4 FreeListInit() . . . 868
9.155 FreeList.h . . . 869

9.156 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/LinkedList.h File Reference 870
9.156.1 Typedef Documentation 872
9.156.1.1 cmp_routine 872
9.156.1.2 free_function 872
9.156.1.3 LinkedList 872
9.156.1.4 ListNode 872
9.156.2 Function Documentation 872
9.156.2.1 ListAddAfter() 872
9.156.2.2 ListAddBefore() 873
9.156.2.3 ListAddHead() 873
9.156.2.4 ListAddTail() 874
9.156.2.5 ListDelNode() 874
9.156.2.6 ListDestroy() 874
9.156.2.7 ListFind() 875
9.156.2.8 ListHead() 875
9.156.2.9 ListInit() 876
9.156.2.10 ListNext() 876
9.156.2.11 ListPrev() 877
9.156.2.12 ListSize() 877
9.156.2.13 ListTail() 877
9.157 LinkedList.h 878
9.158 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.c File Reference 879
9.158.1 Function Documentation 882
9.158.1.1 AddWorker() 882
9.158.1.2 BumpPriority() 882
9.158.1.3 CalcWaitTime() 883
9.158.1.4 CmpThreadPoolJob() 883
9.158.1.5 CreateThreadPoolJob() 883
9.158.1.6 CreateWorker() 884
9.158.1.7 DiffMillis() 884
9.158.1.8 FreeThreadPoolJob() 885
9.158.1.9 SetPolicyType() 885
9.158.1.10 SetPriority() 885
9.158.1.11 SetRelTimeout() 886
9.158.1.12 SetSeed() 886
9.158.1.13 StatsAccountHQ() 886
9.158.1.14 StatsAccountLQ() 886
9.158.1.15 StatsAccountMQ() 887
9.158.1.16 StatsInit() 887
9.158.1.17 StatsTime() 887
9.158.1.18 ThreadPoolAdd() 888
9.158.1.19 ThreadPoolAddPersistent() 888

9.158.1.20 ThreadPoolGetAttr() . . . 889
9.158.1.21 ThreadPoolGetStats() . . . 889
9.158.1.22 ThreadPoolInit() . . . 890
9.158.1.23 ThreadPoolPrintStats() . . . 890
9.158.1.24 ThreadPoolRemove() . . . 891
9.158.1.25 ThreadPoolSetAttr() . . . 891
9.158.1.26 ThreadPoolShutdown() . . . 892
9.158.1.27 TPAttrInit() . . . 892
9.158.1.28 TPAttrSetIdleTime() . . . 892
9.158.1.29 TPAttrSetJobsPerThread() . . . 893
9.158.1.30 TPAttrSetMaxJobsTotal() . . . 893
9.158.1.31 TPAttrSetMaxThreads() . . . 894
9.158.1.32 TPAttrSetMinThreads() . . . 894
9.158.1.33 TPAttrSetSchedPolicy() . . . 895
9.158.1.34 TPAttrSetStackSize() . . . 895
9.158.1.35 TPAttrSetStarvationTime() . . . 895
9.158.1.36 TPJobInit() . . . 896
9.158.1.37 TPJobSetFreeFunction() . . . 896
9.158.1.38 TPJobSetPriority() . . . 897
9.158.1.39 TPSetMaxJobsTotal() . . . 897
9.158.1.40 WorkerThread() . . . 897
9.159 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.h File Reference . . . 898
9.159.1 Macro Definition Documentation . . . 901
9.159.1.1 DEFAULT_FREE_ROUTINE . . . 901
9.159.1.2 DEFAULT_IDLE_TIME . . . 901
9.159.1.3 DEFAULT_JOBS_PER_THREAD . . . 901
9.159.1.4 DEFAULT_MAX_JOBS_TOTAL . . . 901
9.159.1.5 DEFAULT_MAX_THREADS . . . 901
9.159.1.6 DEFAULT_MIN_THREADS . . . 901
9.159.1.7 DEFAULT_PRIORITY . . . 901
9.159.1.8 DEFAULT_STACK_SIZE . . . 902
9.159.1.9 DEFAULT_STARVATION_TIME . . . 902
9.159.1.10 INVALID_JOB_ID . . . 902
9.159.1.11 INVALID_POLICY . . . 902
9.159.1.12 JOBFREELISTSIZE . . . 902
9.159.1.13 STATS . . . 902
9.159.2 Typedef Documentation . . . 902
9.159.2.1 free_routine . . . 902
9.159.2.2 ThreadPool . . . 903
9.159.2.3 ThreadPoolAttr . . . 903
9.159.2.4 ThreadPoolJob . . . 903
9.159.2.5 ThreadPoolStats . . . 903

9.159.3 Function Documentation . . . 903
9.159.3.1 ThreadPoolAdd() . . . 903
9.159.3.2 ThreadPoolAddPersistent() . . . 904
9.159.3.3 ThreadPoolGetAttr() . . . 904
9.159.3.4 ThreadPoolGetStats() . . . 905
9.159.3.5 ThreadPoolInit() . . . 905
9.159.3.6 ThreadPoolPrintStats() . . . 906
9.159.3.7 ThreadPoolRemove() . . . 906
9.159.3.8 ThreadPoolSetAttr() . . . 907
9.159.3.9 ThreadPoolShutdown() . . . 907
9.159.3.10 TPAttrInit() . . . 908
9.159.3.11 TPAttrSetIdleTime() . . . 908
9.159.3.12 TPAttrSetJobsPerThread() . . . 909
9.159.3.13 TPAttrSetMaxJobsTotal() . . . 909
9.159.3.14 TPAttrSetMaxThreads() . . . 909
9.159.3.15 TPAttrSetMinThreads() . . . 910
9.159.3.16 TPAttrSetSchedPolicy() . . . 910
9.159.3.17 TPAttrSetStackSize() . . . 911
9.159.3.18 TPAttrSetStarvationTime() . . . 911
9.159.3.19 TPJobInit() . . . 911
9.159.3.20 TPJobSetFreeFunction() . . . 912
9.159.3.21 TPJobSetPriority() . . . 912
9.159.3.22 TPSetMaxJobsTotal() . . . 913
9.160 ThreadPool.h . . . 913
9.161 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.c File Reference . . . 917
9.161.1 Function Documentation . . . 918
9.161.1.1 CalculateEventTime() . . . 918
9.161.1.2 CreateTimerEvent() . . . 918
9.161.1.3 FreeTimerEvent() . . . 919
9.161.1.4 TimerThreadInit() . . . 919
9.161.1.5 TimerThreadRemove() . . . 919
9.161.1.6 TimerThreadSchedule() . . . 920
9.161.1.7 TimerThreadShutdown() . . . 920
9.161.1.8 TimerThreadWorker() . . . 921
9.162 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.h File Reference . . . 921
9.162.1 Typedef Documentation . . . 923
9.162.1.1 TimeoutType . . . 923
9.162.1.2 TimerEvent . . . 923
9.162.1.3 TimerThread . . . 923
9.162.2 Enumeration Type Documentation . . . 923
9.162.2.1 timeoutType . . . 923
9.162.3 Function Documentation . . . 924

9.162.3.1 TimerThreadInit() . . . 924
9.162.3.2 TimerThreadRemove() . . . 924
9.162.3.3 TimerThreadSchedule() . . . 925
9.162.3.4 TimerThreadShutdown() . . . 925
9.163 TimerThread.h . . . 926
9.164 /build/reproducible-path/pupnp-1.14.20/upnp/src/urlconfig/urlconfig.c File Reference . . . 927
9.165 /build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/sysdep.c File Reference . . . 927
9.165.1 Function Documentation . . . 928
9.165.1.1 get_ieee_node_identifier() . . . 928
9.165.1.2 get_system_time() . . . 928
9.166 /build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/uuid.c File Reference . . . 929
9.166.1 Macro Definition Documentation . . . 930
9.166.1.1 CHECK . . . 930
9.166.2 Function Documentation . . . 930
9.166.2.1 format_uuid_v1() . . . 930
9.166.2.2 format_uuid_v3() . . . 930
9.166.2.3 get_current_time() . . . 931
9.166.2.4 read_state() . . . 931
9.166.2.5 true_random() . . . 931
9.166.2.6 upnp_uuid_unpack() . . . 931
9.166.2.7 uuid_compare() . . . 932
9.166.2.8 uuid_create() . . . 932
9.166.2.9 uuid_create_from_name() . . . 932
9.166.2.10 write_state() . . . 933

Index **935**

Chapter 1

API changes after 1.10

This is a list of API changes after libUPnP version 1.10. This page contains explanation of these changes and what users need to do to migrate.

1.1 Version 1.12

1.1.1 list.h header changed

The `list.h` header was replace by a different version, now all symbols are properly namespaced with a Upnp prefix. This header was never intended to be used directly by libUPnP users, so not many should be affected.

As a side-effect of the header change, libUPnP no longer defines the `true` and `bool` macros. Users that need those should define their own or use `<stdbool.h>`. Additionally the following macros are no longer defined:

- `prefetch`
- `container_of`
- `READ_ONCE`
- `WRITE_ONCE`

And various list-specific macros are gone. For details see the discussion at the Github PRs `#128` and `#129`.

1.1.2 Removal of TRUE, FALSE macros and BOOL type

Previously `ixml.h` used to define `TRUE` and `FALSE` and declared a `BOOL` typedef. Those macros and the typedef were removed in 10.12 in favor of just using `int` and `1` and `0` for true and false.

If your code relied on these being available, the cleanets fix would be to use `<stdbool.h>` instead. Alternatively if you can't use that or it would require too many changes to your existing source code, you can define the macros and typedef in your headers:

```
#ifndef __OBJC__
    typedef int BOOL;
#else
    /* When using Objective-C, include objc.h which defines BOOL */
    #include <objc/objc.h>
#endif

#define TRUE 1
#define FALSE 0
```

1.2 Version 1.14

1.2.1 UpnpInit() has been dropped

UpnpInit() has long been deprecated and has now been dropped. Please use UpnpInit2().

1.2.2 Some include files have been renamed

When the template code was removed, a few files were renamed for consistency. The function calls remain the same, so the only necessary changes are in the include file names. The affected files are:

Original Name	New Name
upnp/inc/ActionComplete.h	upnp/inc/UpnpActionComplete.h
upnp/inc/ActionRequest.h	upnp/inc/UpnpActionRequest.h
upnp/inc/Discovery.h	upnp/inc/UpnpDiscovery.h
upnp/inc/Event.h	upnp/inc/UpnpEvent.h
upnp/inc/EventSubscribe.h	upnp/inc/UpnpEventSubscribe.h
upnp/inc/ExtraHeaders.h	upnp/inc/UpnpExtraHeaders.h
upnp/inc/FileInfo.h	upnp/inc/UpnpFileInfo.h
upnp/inc/StateVarComplete.h	upnp/inc/UpnpStateVarComplete.h
upnp/inc/StateVarRequest.h	upnp/inc/UpnpStateVarRequest.h
upnp/inc/Subscription↩ Request.h	upnp/inc/UpnpSubscriptionRequest.h

Chapter 2

Deprecated List

Global UpnpGetServiceVarStatus (UpnpClient_Handle Hnd, const char *ActionURL, const char *VarName, DOMString *StVarVal)

The use of this function is deprecated by the UPnP Forum.

Global UpnpGetServiceVarStatusAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *↩ VarName, Upnp_FunPtr Fun, const void *Cookie)

The use of this function is deprecated by the UPnP Forum.

Global UpnpSetContentLength (UpnpClient_Handle Hnd, size_t contentLength)

Use **UpnpSetMaxContentLength** instead.

Chapter 3

Todo List

Global ixmlDocument_setOwnerDocument (IXML_Document *doc, IXML_Node *nodeptr)

documentation.

Global ixmlDomTreetoString (IXML_Node *nodeptr, ixml_membuf *buf)

documentation.

documentation.

Global ixmlPrintDomTree (IXML_Node *nodeptr, ixml_membuf *buf)

documentation.

documentation.

Global ixmlPrintDomTreeRecursive (IXML_Node *nodeptr, ixml_membuf *buf)

documentation.

documentation.

Global UpnpResolveURL (const char *BaseURL, const char *RelURL, char *AbsURL)

There is some unnecessary allocation and deallocation going on here because of the way resolve_rel_url() was originally written and used. In the future it would be nice to clean this up.

Module UpnpString

Always alloc a minimum size like 64 bytes or so and when shrinking do not perform a new memory allocation.

Chapter 4

Topic Index

4.1 Topics

Here is a list of all topics with brief descriptions:

UPnP API 17
The UpnpString Class 97
Optional Tool API 105
Sample Code 112
Network Socket Library 154
SSDP Library 160
XML API 176

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

_IXML_ATTR
Data structure representing an Attribute node . . . 223
_IXML_CDATASection
Data structure representing a CDATA section node . . . 224
_IXML_Document
Data structure representing the DOM Document . . . 225
_IXML_Element
Data structure representing an Element node . . . 225
_IXML_ElementStack . . . 226
_IXML_NamedNodeMap
Data structure representing a list of named nodes . . . 227
_IXML_NamespaceURI . . . 228
_IXML_Node
Data structure common to all types of nodes . . . 228
_IXML_NodeList
Data structure representing a list of nodes . . . 229
_IXML_Text
Data structure representing a Text node . . . 230
_Parser . . . 231
_uuid_upnp . . . 232
char_info . . . 233
cmdloop_commands . . . 233
document_type_t . . . 234
ErrorString
Structure to maintain a error code and string associated with the error code . . . 234
FREELIST . . . 235
FREELISTNODE . . . 236
Handle_Info . . . 237
HOSTPORT
Represents a host port: e.g. "127.127.0.1:80" text is a token pointing to the full string representation . . . 241
http_header_t . . . 242
http_message_t . . . 243
http_parser_t . . . 246
HTTPCONNECTIONHANDLE . . . 247

ixml_membuf
The ixml_membuf type . . . 248
job_arg . . . 249
LINKEDLIST . . . 250
LISTNODE . . . 252
MD5Context . . . 253
membuffer . . . 253
memptr . . . 254
mserv_request_t . . . 255
MServerSockArray . . . 256
NOTIFY_THREAD_STRUCT . . . 258
s_Class . . . 258
s_GenlibClientSubscription . . . 259
s_Member . . . 259
s_SSDPResultData . . . 260
s_TestClass . . . 260
s_Type_Integer . . . 261
s_UpnpActionComplete . . . 261
s_UpnpActionRequest . . . 262
s_UpnpDiscovery . . . 263
s_UpnpEvent . . . 263
s_UpnpEventSubscribe . . . 264
s_UpnpExtraHeaders . . . 264
s_UpnpFileInfo . . . 265
s_UpnpStateVarComplete . . . 265
s_UpnpStateVarRequest . . . 266
s_UpnpSubscriptionRequest . . . 266
scanner_t . . . 266
SendInstruction . . . 267
SERVICE_INFO . . . 269
SERVICE_TABLE . . . 270
soap_devserv_t . . . 271
SOCKINFO . . . 271
ssdp_thread_data . . . 272
SsdpEventStruct . . . 272
ssdpsearcharg . . . 273
ssdpsearchexparg . . . 273
ssdpsearchreply . . . 274
SSDPSockArray . . . 274
str_int_entry . . . 274
SUBSCRIPTION . . . 275
SUpnpString
Internal implementation of the class UpnpString . . . 275
TData . . . 276
TheStruct . . . 276
THREADPOOL
A thread pool similar to the thread pool in the UPnP SDK . . . 277
THREADPOOLATTR . . . 281
THREADPOOLJOB . . . 282
TIMEREVENT . . . 283
TIMERTHREAD . . . 284
TOKEN
Buffer used in parsinghttp messages, urls, etc. generally this simply holds a pointer into a larger array . . . 285
TPOOLSTATS . . . 285
tv_service . . . 286
TvDevice . . . 287
TvDeviceNode . . . 287

TvService 288
UpnpListHead 290
UpnpNonblockParam 291
URI
Represents a URI used in parse_uri and elsewhere 292
URL_LIST
Represents a list of URLs as in the "callback" header of SUBSCRIBE message in GENA. "char ∗" URLs holds dynamic memory 293
uuid_node_t 294
uuid_state 294
virtual_Dir_List 295
VirtualDirCallbacks 296
xml_alias_t 298

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

/build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h . . . 301
/build/reproducible-path/pupnp-1.14.20/ixml/inc/ixmldebug.h
Auxiliar routines to aid debugging . . . 314
/build/reproducible-path/pupnp-1.14.20/ixml/inc/posix_overwrites.h . . . 317
/build/reproducible-path/pupnp-1.14.20/ixml/src/document.c . . . 318
/build/reproducible-path/pupnp-1.14.20/ixml/src/element.c . . . 319
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixml.c . . . 337
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixmldebug.c . . . 340
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlmembuf.c . . . 342
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlparser.c . . . 347
/build/reproducible-path/pupnp-1.14.20/ixml/src/namedNodeMap.c . . . 371
/build/reproducible-path/pupnp-1.14.20/ixml/src/node.c . . . 373
/build/reproducible-path/pupnp-1.14.20/ixml/src/nodeList.c . . . 384
/build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlmembuf.h . . . 322
/build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlparser.h . . . 328
/build/reproducible-path/pupnp-1.14.20/upnp/generator/generator.h . . . 386
/build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.c
Source file for TestClass methods . . . 387
/build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.h
Header file for TestClass methods . . . 394
/build/reproducible-path/pupnp-1.14.20/upnp/generator/TheStruct.h . . . 402
/build/reproducible-path/pupnp-1.14.20/upnp/inc/Callback.h . . . 402
/build/reproducible-path/pupnp-1.14.20/upnp/inc/ithread.h . . . 406
/build/reproducible-path/pupnp-1.14.20/upnp/inc/list.h . . . 418
/build/reproducible-path/pupnp-1.14.20/upnp/inc/upnp.h . . . 419
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionComplete.h
Header file for UpnpActionComplete methods . . . 435
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionRequest.h
Header file for UpnpActionRequest methods . . . 441
/build/reproducible-path/pupnp-1.14.20/upnp/inc/upnpdebug.h . . . 455
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpDiscovery.h
Header file for UpnpDiscovery methods . . . 461
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEvent.h
Header file for UpnpEvent methods . . . 479
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEventSubscribe.h
Header file for UpnpEventSubscribe methods . . . 484

/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpExtraHeaders.h
Header file for UpnpExtraHeaders methods . . . 491
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpFileInfo.h
Header file for UpnpFileInfo methods . . . 499
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpGlobal.h
Defines constants that for some reason are not defined on some systems . . . 507
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpInet.h
Provides a platform independent way to include TCP/IP types and functions . . . 510
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpIntTypes.h . . . 512
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarComplete.h
Header file for UpnpStateVarComplete methods . . . 512
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarRequest.h
Header file for UpnpStateVarRequest methods . . . 519
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStdInt.h . . . 532
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpString.h
UpnpString object declaration . . . 532
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpSubscriptionRequest.h
Header file for UpnpSubscriptionRequest methods . . . 535
/build/reproducible-path/pupnp-1.14.20/upnp/inc/upnptools.h . . . 543
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpUniStd.h . . . 546
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util.c . . . 546
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util.h . . . 547
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.c . . . 551
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.h . . . 553
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.c . . . 558
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.h . . . 560
/build/reproducible-path/pupnp-1.14.20/upnp/src/inet_pton.c . . . 844
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionComplete.c
Source file for UpnpActionComplete methods . . . 567
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionRequest.c
Source file for UpnpActionRequest methods . . . 572
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpapi.c . . . 584
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpdebug.c . . . 590
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpDiscovery.c
Source file for UpnpDiscovery methods . . . 594
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEvent.c
Source file for UpnpEvent methods . . . 610
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEventSubscribe.c
Source file for UpnpEventSubscribe methods . . . 614
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpExtraHeaders.c
Source file for UpnpExtraHeaders methods . . . 620
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpFileInfo.c
Source file for UpnpFileInfo methods . . . 626
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarComplete.c
Source file for UpnpStateVarComplete methods . . . 633
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarRequest.c
Source file for UpnpStateVarRequest methods . . . 639
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpString.c
UpnpString object implementation . . . 650
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpSubscriptionRequest.c
Source file for UpnpSubscriptionRequest methods . . . 651
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnptools.c . . . 658
/build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_ctrlpt.c . . . 661
/build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_device.c . . . 667
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/client_table/GenlibClientSubscription.c
Source file for GenlibClientSubscription methods . . . 677
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/miniserver/miniserver.c
Implements the functionality and utility functions used by the Miniserver module . . . 685

/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/sock.c
Implements the sockets functionality . . . 718
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/httpreadwrite.c . . . 690
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/parsetools.c . . . 702
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/webserver.c
Defines the Web Server and has functions to carry out operations of the Web Server . . . 703
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/uri/uri.c
Contains functions for uri, url parsing utility . . . 719
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/util/util.c . . . 729
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/client_table.h . . . 731
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/config.h . . . 735
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena.h . . . 737
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_ctrlpt.h . . . 749
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_device.h . . . 750
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/GenlibClientSubscription.h
Header file for GenlibClientSubscription methods . . . 753
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h . . . 763
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpreadwrite.h . . . 771
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/inet_pton.h . . . 775
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/md5.h . . . 775
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/membuffer.h . . . 776
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/miniserver.h . . . 784
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/netall.h . . . 789
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/parsetools.h . . . 789
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/service_table.h . . . 791
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/soaplib.h . . . 803
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sock.h . . . 806
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h . . . 809
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/statcodes.h . . . 816
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/strintmap.h . . . 817
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sysdep.h . . . 818
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/unixutil.h . . . 821
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnpapi.h . . . 822
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnputil.h . . . 828
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h . . . 831
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/urlconfig.h . . . 841
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uuid.h . . . 842
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/VirtualDir.h . . . 843
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/webserver.h . . . 843
/build/reproducible-path/pupnp-1.14.20/upnp/src/soap/soap_device.c . . . 844
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_ctrlpt.c . . . 851
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_device.c . . . 852
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_server.c . . . 854
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.c
Source file for SSDPResultData methods . . . 855
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.h
Header file for SSDPResultData methods . . . 859
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultDataCallback.c
SSDPResultDataCallback . . . 863
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultDataCallback.h
SSDPResultData object declararion . . . 864
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/FreeList.h . . . 866
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/LinkedList.h . . . 870
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.c . . . 879
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.h . . . 898
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.c . . . 917
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.h . . . 921
/build/reproducible-path/pupnp-1.14.20/upnp/src/urlconfig/urlconfig.c . . . 927

/build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/sysdep.c 927
/build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/uuid.c 929

Chapter 7

Topic Documentation

7.1 UPnP API

Files

- file upnp.h
- file upnpapi.c

Data Structures

- struct job_arg

Macros

- #define **LINE_SIZE** (size_t)180
- #define **NAME_SIZE** (size_t)256
- #define **MNFT_NAME_SIZE** 64
- #define **MODL_NAME_SIZE** 32
- #define **SERL_NUMR_SIZE** 64
- #define **MODL_DESC_SIZE** 64
- #define **UPNP_INFINITE** -1
- #define **UPNP_USING_CHUNKED** -3
- #define **UPNP_UNTIL_CLOSE** -4
- #define **ifr_netmask** ifr_addr
- #define IN6_IS_ADDR_GLOBAL(a)
- #define IN6_IS_ADDR_ULA(a)

Functions

- static void free_advertise_arg (job_arg *arg)

  Free memory associated with advertise job's argument.
- static void free_action_arg (job_arg *arg)

  Free memory associated with an action job's argument.
- static int WinsockInit (void)

  (Windows Only) Initializes the Windows Winsock library.
- static int UpnpInitMutexes (void)

  Initializes the global mutexes used by the UPnP SDK.
- static int UpnpInitThreadPools (void)

  Initializes the global threadm pools used by the UPnP SDK.
- static int UpnpInitPreamble (void)

  Performs the initial steps in initializing the UPnP SDK.
- static int UpnpInitStartServers (unsigned short DestPort)

  Finishes initializing the UPnP SDK.
- void PrintThreadPoolStats (ThreadPool *tp, const char *DbgFileName, int DbgLineNo, const char *msg)

  Prints thread pool statistics.
- static int GetFreeHandle ()

  Get a free handle.
- static int FreeHandle (int Upnp_Handle)

  Free handle.
- static int GetDescDocumentAndURL (Upnp_DescType descriptionType, char *description, int config_base↩ URL, int AddressFamily, IXML_Document **xmlDoc, char descURL[LINE_SIZE])

  Fills the sockadr_in with miniserver information.
- static unsigned UpnpComputeIpv6PrefixLength (struct sockaddr_in6 *Netmask)

  Computes prefix length from IPv6 netmask.
- int UpnpGetIfInfo (const char *IfName)

  Retrieve interface information and keep it in global variables. If NULL, we'll find the first suitable interface for operation.
- void UpnpThreadDistribution (struct UpnpNonblockParam *Param)

  Schedule async functions in threadpool.
- Upnp_FunPtr GetCallBackFn (UpnpClient_Handle Hnd)

  Get callback function ptr from a handle.
- Upnp_Handle_Type GetClientHandleInfo (UpnpClient_Handle *client_handle_out, struct Handle_Info **HndInfo)

  Get client handle info.
- Upnp_Handle_Type GetDeviceHandleInfo (UpnpDevice_Handle start, int AddressFamily, UpnpDevice_Handle *device_handle_out, struct Handle_Info **HndInfo)

  Retrieves the device handle and information of the first device of the address family specified. The search begins at the 'start' index, which should be 0 for the first call, then the last successful value returned. This allows listing all entries for the address family.
- Upnp_Handle_Type GetDeviceHandleInfoForPath (const char *path, int AddressFamily, UpnpDevice_Handle *device_handle_out, struct Handle_Info **HndInfo, service_info **serv_info)

  Retrieves the device handle and information of the first device of the address family specified, with a service having a controlURL or eventSubURL matching the path.
- Upnp_Handle_Type GetHandleInfo (UpnpClient_Handle Hnd, struct Handle_Info **HndInfo)

  Get handle information.
- int PrintHandleInfo (UpnpClient_Handle Hnd)

  Print handle info.
- void AutoAdvertise (void *input)

  This function is a timer thread scheduled by UpnpSendAdvertisement to the send advetisement again.
- int **UpnpSetEventQueueLimits** (int maxLen, int maxAge)

Variables

- struct VirtualDirCallbacks virtualDirCallback
- virtualDirList * pVirtualDirList
- ithread_mutex_t GlobalClientSubscribeMutex
- ithread_rwlock_t GlobalHndRWLock
- ithread_mutex_t gUUIDMutex
- ithread_mutex_t gSDKInitMutex = PTHREAD_MUTEX_INITIALIZER
- TimerThread gTimerThread
- ThreadPool gSendThreadPool
- ThreadPool gRecvThreadPool
- ThreadPool gMiniServerThreadPool
- WebServerState bWebServerState = WEB_SERVER_DISABLED
- WebCallback_HostValidate gWebCallback_HostValidate = 0
- void * gWebCallback_HostValidateCookie = 0
- int gAllowLiteralHostRedirection = 0
- char gIF_NAME [LINE_SIZE] = {'\0'}
- char gIF_IPV4 [INET_ADDRSTRLEN] = {'\0'}
- char gIF_IPV4_NETMASK [INET_ADDRSTRLEN] = {'\0'}
- char gIF_IPV6 [INET6_ADDRSTRLEN] = {'\0'}
- unsigned gIF_IPV6_PREFIX_LENGTH = 0
- char gIF_IPV6_ULA_GUA [INET6_ADDRSTRLEN] = {'\0'}
- unsigned gIF_IPV6_ULA_GUA_PREFIX_LENGTH = 0
- unsigned gIF_INDEX = (unsigned)-1
- unsigned short LOCAL_PORT_V4
- unsigned short LOCAL_PORT_V6
- unsigned short LOCAL_PORT_V6_ULA_GUA
- static void * HandleTable [NUM_HANDLE]
- membuffer gDocumentRootDir
- membuffer gWebserverCorsString
- size_t g_maxContentLength = DEFAULT_SOAP_CONTENT_LENGTH
- int g_UpnpSdkEQMaxLen = MAX_SUBSCRIPTION_QUEUED_EVENTS
- int g_UpnpSdkEQMaxAge = MAX_SUBSCRIPTION_EVENT_AGE
- int UpnpSdkInit = 0
- int UpnpSdkClientRegistered = 0
- int UpnpSdkDeviceRegisteredV4 = 0
- int UpnpSdkDeviceregisteredV6 = 0
- int **job_arg::handle**
- int **job_arg::eventId**
- void * **job_arg::Event**
- struct {
  int **handle**
  int **eventId**
  void * **Event**
  } **job_arg::advertise**

- struct UpnpNonblockParam **job_arg::action**

Constants and Types

- enum **UpnpOpenFileMode** { **UPNP_READ** , **UPNP_WRITE** }
- enum Upnp_SType_e { UPNP_S_ALL , UPNP_S_ROOT , UPNP_S_DEVICE , UPNP_S_SERVICE }

  Represents the different types of searches that can be performed using the SDK for UPnP Devices API.
- enum Upnp_DescType_e { UPNPREG_URL_DESC , UPNPREG_FILENAME_DESC , UPNPREG_BUF_DESC }

  *Specifies the type of description in **UpnpRegisterRootDevice2**.*
- typedef int UpnpClient_Handle

  *Returned when a control point application registers with **UpnpRegisterClient**.*
- typedef int UpnpDevice_Handle

  *Returned when a device application registers with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **Upnp↩ RegisterRootDevice3** or **UpnpRegisterRootDevice4**.*
- typedef char Upnp_SID[44]

  Holds the subscription identifier for a subscription between a client and a device.
- typedef enum Upnp_SType_e **Upnp_SType**
- typedef enum Upnp_DescType_e **Upnp_DescType**

Control Point HTTP API

- enum Upnp_HttpMethod_e {
  UPNP_HTTPMETHOD_PUT = 0 , **UPNP_HTTPMETHOD_DELETE** = 1 , **UPNP_HTTPMETHOD_GET** = 2 ,
  UPNP_HTTPMETHOD_HEAD = 3 ,
  UPNP_HTTPMETHOD_POST = 4 }

  Different HTTP methods.
- typedef enum Upnp_HttpMethod_e **Upnp_HttpMethod**
- int UpnpDownloadUrlItem (const char ∗url, char ∗∗outBuf, char ∗contentType)

  Downloads a file specified in a URL.
- int UpnpOpenHttpGet (const char ∗url, void ∗∗handle, char ∗∗contentType, int ∗contentLength, int ∗http↩ Status, int timeout)

  Gets a file specified in a URL.
- int UpnpOpenHttpGetProxy (const char ∗url, const char ∗proxy_str, void ∗∗handle, char ∗∗contentType, int ∗contentLength, int ∗httpStatus, int timeout)

  Gets a file specified in a URL through the specified proxy.
- int UpnpOpenHttpGetEx (const char ∗url, void ∗∗handle, char ∗∗contentType, int ∗contentLength, int ∗http↩ Status, int lowRange, int highRange, int timeout)

  Gets specified number of bytes from a file specified in the URL.
- int UpnpReadHttpGet (void ∗handle, char ∗buf, size_t ∗size, int timeout)

  Gets specified number of bytes from a file specified in a URL.
- int UpnpHttpGetProgress (void ∗handle, size_t ∗length, size_t ∗total)

  Retrieve progress information of a http-get transfer.
- int UpnpCancelHttpGet (void ∗handle)

  *Set the cancel flag of the **handle** parameter.*
- int UpnpCloseHttpGet (void ∗handle)

  *Closes the connection and frees memory that was allocated for the **handle** parameter.*
- int UpnpOpenHttpPost (const char ∗url, void ∗∗handle, const char ∗contentType, int contentLength, int timeout)

  Makes an HTTP POST request message, opens a connection to the server and sends the POST request to the server if the connection to the server succeeds.
- int UpnpWriteHttpPost (void ∗handle, char ∗buf, size_t ∗size, int timeout)

  *Sends a request to a server to copy the contents of a buffer to the URI specified in the **UpnpOpenHttpPost** call.*
- int UpnpCloseHttpPost (void ∗handle, int ∗httpStatus, int timeout)

*Sends and receives any pending data, closes the connection with the server, and frees memory allocated during the **UpnpOpenHttpPost** call.*

- int UpnpOpenHttpConnection (const char *url, void **handle, int timeout)

  Opens a connection to the server.
- int UpnpMakeHttpRequest (Upnp_HttpMethod method, const char *url, void *handle, UpnpString *headers, const char *contentType, int contentLength, int timeout)

  *Makes a HTTP request using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpWriteHttpRequest (void *handle, char *buf, size_t *size, int timeout)

  *Writes the content of a HTTP request initiated by a **UpnpMakeHttpRequest** call. The end of the content should be indicated by a call to **UpnpEndHttpRequest**.*
- int UpnpEndHttpRequest (void *handle, int timeout)

  *Indicates the end of a HTTP request previously made by **UpnpMakeHttpRequest**.*
- int UpnpGetHttpResponse (void *handle, UpnpString *headers, char **contentType, int *contentLength, int *httpStatus, int timeout)

  *Gets the response from the server using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpReadHttpResponse (void *handle, char *buf, size_t *size, int timeout)

  *Reads the content of a response using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpCloseHttpConnection (void *handle)

  *Closes the connection created with **UpnpOpenHttpConnection** and frees any memory associated with the connection.*
- int UpnpDownloadXmlDoc (const char *url, IXML_Document **xmlDoc)

  Downloads an XML document specified in a URL.

Web Server API

- typedef void * UpnpWebFileHandle

  The type of handle returned by the web server for open requests.
- typedef int(* VDCallback_GetInfo) (const char *filename, UpnpFileInfo *info, const void *cookie, const void **request_cookie)

  Get-info callback function prototype.
- typedef UpnpWebFileHandle(* VDCallback_Open) (const char *filename, enum UpnpOpenFileMode Mode, const void *cookie, const void *request_cookie)

  Open callback function prototype.
- typedef int(* VDCallback_Read) (UpnpWebFileHandle fileHnd, char *buf, size_t buflen, const void *cookie, const void *request_cookie)

  Read callback function prototype.
- typedef int(* VDCallback_Write) (UpnpWebFileHandle fileHnd, char *buf, size_t buflen, const void *cookie, const void *request_cookie)

  Write callback function prototype.
- typedef int(* VDCallback_Seek) (UpnpWebFileHandle fileHnd, off_t offset, int origin, const void *cookie, const void *request_cookie)

  Seek callback function prototype.
- typedef int(* VDCallback_Close) (UpnpWebFileHandle fileHnd, const void *cookie, const void *request_↩ cookie)

  Close callback function prototype.
- typedef int(* **WebCallback_HostValidate**) (const char *hostname, void *cookie)
- int UpnpSetWebServerRootDir (const char *rootDir)

  Sets the document root directory for the internal web server.
- int UpnpVirtualDir_set_GetInfoCallback (VDCallback_GetInfo callback)

  Sets the get_info callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_OpenCallback (VDCallback_Open callback)

  Sets the open callback function to be used to access a virtual directory.

- int UpnpVirtualDir_set_ReadCallback (VDCallback_Read callback)

  Sets the read callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_WriteCallback (VDCallback_Write callback)

  Sets the write callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_SeekCallback (VDCallback_Seek callback)

  Sets the seek callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_CloseCallback (VDCallback_Close callback)

  Sets the close callback function to be used to access a virtual directory.
- int UpnpEnableWebserver (int enable)

  Enables or disables the webserver.
- int UpnpIsWebserverEnabled (void)

  Returns `1` if the webserver is enabled, or `0` if it is not.
- void **UpnpSetHostValidateCallback** (WebCallback_HostValidate callback, void *cookie)
- void **UpnpSetAllowLiteralHostRedirection** (int enable)
- int UpnpSetWebServerCorsString (const char *corsString)

  Assign the Access-Control-Allow-Origin specfied by the input const char cors_string parameterto the global CORS string.*
- int UpnpAddVirtualDir (const char *dirName, const void *cookie, const void **oldcookie)

  Adds a virtual directory mapping.
- int UpnpRemoveVirtualDir (const char *dirName)

  *Removes a virtual directory mapping made with **UpnpAddVirtualDir**.*
- void UpnpRemoveAllVirtualDirs (void)

  Removes all virtual directory mappings.

Initialization and Registration

- int UpnpInit2 (const char *IfName, unsigned short DestPort)

  Initializes the Linux SDK for UPnP Devices (IPv4 or IPv6).
- int UpnpFinish (void)

  Initializes the OpenSSL library, and the OpenSSL context for use with pupnp.
- unsigned short UpnpGetServerPort (void)

  Returns the internal server IPv4 UPnP listening port.
- unsigned short UpnpGetServerPort6 (void)

  Returns the internal server IPv6 link-local (LLA) UPnP listening port.
- unsigned short UpnpGetServerUlaGuaPort6 (void)

  Returns the internal server IPv6 ULA or GUA UPnP listening port.
- char * UpnpGetServerIpAddress (void)

  Returns the local IPv4 listening ip address.
- char * UpnpGetServerIp6Address (void)

  Returns the IPv6 link-local listening ip address.
- char * UpnpGetServerUlaGuaIp6Address (void)

  Returns the IPv6 unique-local or globally-unique listening ip address.
- int UpnpRegisterRootDevice (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd)

  Registers a device application with the UPnP Library.
- int UpnpRegisterRootDevice2 (Upnp_DescType descriptionType, const char *description, size_t bufferLen, int config_baseURL, Upnp_FunPtr Fun, const void *Cookie, UpnpDevice_Handle *Hnd)

  *Registers a device application with the UPnP Library. Similar to **UpnpRegisterRootDevice**, except that it also allows the description document to be specified as a file or a memory buffer.*
- int UpnpRegisterRootDevice3 (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd, int AddressFamily)

Registers a device application for a specific address family with the UPnP library.

- int UpnpRegisterRootDevice4 (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd, int AddressFamily, const char *LowerDescUrl)

  Registers a device application for a specific address family with the UPnP library. This function can also be used to specify a dedicated description URL to be returned for legacy CPs.
- int UpnpUnRegisterRootDevice (UpnpDevice_Handle Hnd)

  *Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **UpnpRegisterRootDevice3** or **UpnpRegisterRootDevice4**.*
- int UpnpUnRegisterRootDeviceLowPower (UpnpDevice_Handle Hnd, int PowerState, int SleepPeriod, int RegistrationState)

  *Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **UpnpRegisterRootDevice3** or **UpnpRegisterRootDevice4**.*
- int UpnpRegisterClient (Upnp_FunPtr Callback, const void *Cookie, UpnpClient_Handle *Hnd)

  Registers a control point application with the UPnP Library.
- int UpnpUnRegisterClient (UpnpClient_Handle Hnd)

  Unregisters a control point application, unsubscribing all active subscriptions.
- int UpnpSetContentLength (UpnpClient_Handle Hnd, size_t contentLength)
- int UpnpSetMaxContentLength (size_t contentLength)

  Sets the maximum content-length that the SDK will process on an incoming SOAP requests or responses.

Discovery

- int UpnpSearchAsync (UpnpClient_Handle Hnd, int Mx, const char *TTarget_constarget_const, const void *Cookie_const)

  Searches for devices matching the given search target.
- int UpnpSendAdvertisement (UpnpDevice_Handle Hnd, int Exp)

  Sends out the discovery announcements for all devices and services for a device.
- int UpnpSendAdvertisementLowPower (UpnpDevice_Handle Hnd, int Exp, int PowerState, int SleepPeriod, int RegistrationState)

  Sends out the discovery announcements for all devices and services for a device.

Control

- int UpnpGetServiceVarStatus (UpnpClient_Handle Hnd, const char *ActionURL, const char *VarName, DOMString *StVarVal)

  Queries the state of a state variable of a service on another device.
- int UpnpGetServiceVarStatusAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *VarName, Upnp_FunPtr Fun, const void *Cookie)

  Queries the state of a variable of a service, generating a callback when the operation is complete.
- int UpnpSendAction (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Action, IXML_Document **RespNode)

  Sends a message to change a state variable in a service.
- int UpnpSendActionEx (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Header, IXML_Document *Action, IXML_Document **RespNode)

  Sends a message to change a state variable in a service.
- int UpnpSendActionAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Action, Upnp_FunPtr Fun, const void *Cookie)

  Sends a message to change a state variable in a service, generating a callback when the operation is complete.
- int UpnpSendActionExAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Header, IXML_Document *Action, Upnp_FunPtr Fun, const void *Cookie)

  Sends a message to change a state variable in a service, generating a callback when the operation is complete.

Eventing

- int UpnpAcceptSubscription (UpnpDevice_Handle Hnd, const char *DevID, const char *ServID, const char **VarName, const char **NewVal, int cVariables, const Upnp_SID SubsId)

  Accepts a subscription request and sends out the current state of the eventable variables for a service.
- int UpnpAcceptSubscriptionExt (UpnpDevice_Handle Hnd, const char *DevID, const char *ServID, IXML_Document *PropSet, const Upnp_SID SubsId)

  *Similar to **UpnpAcceptSubscription()** except that it takes a DOM document for the variables to event rather than an array of strings.*
- int UpnpNotify (UpnpDevice_Handle, const char *DevID, const char *ServID, const char **VarName, const char **NewVal, int cVariables)

  Sends out an event change notification to all control points subscribed to a particular service.
- int UpnpNotifyExt (UpnpDevice_Handle, const char *DevID, const char *ServID, IXML_Document *PropSet)

  *Similar to **UpnpNotify** except that it takes a DOM document for the event rather than an array of strings.*
- int UpnpRenewSubscription (UpnpClient_Handle Hnd, int *TimeOut, const Upnp_SID SubsId)

  Renews a subscription that is about to expire.
- int UpnpRenewSubscriptionAsync (UpnpClient_Handle Hnd, int TimeOut, Upnp_SID SubsId, Upnp_FunPtr Fun, const void *Cookie)

  Renews a subscription that is about to expire, generating a callback when the operation is complete.
- int UpnpSetMaxSubscriptions (UpnpDevice_Handle Hnd, int MaxSubscriptions)

  Sets the maximum number of subscriptions accepted per service.
- int UpnpSetMaxSubscriptionTimeOut (UpnpDevice_Handle Hnd, int MaxSubscriptionTimeOut)

  Sets the maximum time-out accepted for a subscription request or renewal.
- int UpnpSubscribe (UpnpClient_Handle Hnd, const char *PublisherUrl, int *TimeOut, Upnp_SID SubsId)

  Registers a control point to receive event notifications from another device.
- int UpnpSubscribeAsync (UpnpClient_Handle Hnd, const char *PublisherUrl, int TimeOut, Upnp_FunPtr Fun, const void *Cookie)

  *Performs the same operation as **UpnpSubscribe**, but returns immediately and calls the registered callback function when the operation is complete.*
- int UpnpUnSubscribe (UpnpClient_Handle Hnd, const Upnp_SID SubsId)

  *Removes the subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**.*
- int UpnpUnSubscribeAsync (UpnpClient_Handle Hnd, Upnp_SID SubsId, Upnp_FunPtr Fun, const void *Cookie)

  *Removes a subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**, generating a callback when the operation is complete.*

Error codes

The functions in the SDK API can return a variety of error codes to describe problems encountered during execution. This section lists the error codes and provides a brief description of what each error code means. Refer to the documentation for each function for a description of what an error code means in that context.

- #define UPNP_E_SUCCESS 0

  The operation completed successfully.
- #define UPNP_E_INVALID_HANDLE -100

  The handle passed to a function is not a recognized as a valid handle.
- #define UPNP_E_INVALID_PARAM -101

  One or more of the parameters passed to the function is not valid.
- #define UPNP_E_OUTOF_HANDLE -102

  The SDK does not have any more space for additional handles.

- #define **UPNP_E_OUTOF_CONTEXT** -103
- #define UPNP_E_OUTOF_MEMORY -104

  Not enough resources are currently available to complete the operation.
- #define UPNP_E_INIT -105

  The SDK has already been initialized.
- #define **UPNP_E_BUFFER_TOO_SMALL** -106
- #define UPNP_E_INVALID_DESC -107

  *The description document passed to **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2** **UpnpRegisterRootDevice3** or **UpnpRegisterRootDevice4** is invalid.*
- #define UPNP_E_INVALID_URL -108

  An URL passed into the function is invalid.
- #define **UPNP_E_INVALID_SID** -109
- #define **UPNP_E_INVALID_DEVICE** -110
- #define UPNP_E_INVALID_SERVICE -111

  The device ID/service ID pair does not refer to a valid service.
- #define UPNP_E_BAD_RESPONSE -113

  The response received from the remote side of a connection is not correct for the protocol.
- #define **UPNP_E_BAD_REQUEST** -114
- #define UPNP_E_INVALID_ACTION -115

  The SOAP action message is invalid.
- #define UPNP_E_FINISH -116

  ***UpnpInit2** has not been called, or **UpnpFinish** has already been called.*
- #define UPNP_E_INIT_FAILED -117

  ***UpnpInit2** cannot complete.*
- #define UPNP_E_URL_TOO_BIG -118

  The URL passed into a function is too long.
- #define UPNP_E_BAD_HTTPMSG -119

  The HTTP message contains invalid message headers.
- #define UPNP_E_ALREADY_REGISTERED -120

  A client or a device is already registered.
- #define UPNP_E_INVALID_INTERFACE -121

  *The interface provided to **UpnpInit2** is unknown or does not have a valid IPv4 or IPv6 address configured.*
- #define UPNP_E_NETWORK_ERROR -200

  A network error occurred.
- #define UPNP_E_SOCKET_WRITE -201

  An error happened while writing to a socket.
- #define UPNP_E_SOCKET_READ -202

  An error happened while reading from a socket.
- #define UPNP_E_SOCKET_BIND -203

  The SDK had a problem binding a socket to a network interface.
- #define UPNP_E_SOCKET_CONNECT -204

  The SDK had a problem connecting to a remote host.
- #define UPNP_E_OUTOF_SOCKET -205

  The SDK cannot create any more sockets.
- #define UPNP_E_LISTEN -206

  The SDK had a problem setting the socket to listen for incoming connections.
- #define UPNP_E_TIMEDOUT -207

  Too much time elapsed before the required number of bytes were sent or received over a socket.
- #define UPNP_E_SOCKET_ERROR -208

  Generic socket error code for conditions not covered by other error codes.
- #define **UPNP_E_FILE_WRITE_ERROR** -209

- #define UPNP_E_CANCELED -210

  The operation was canceled.
- #define **UPNP_E_EVENT_PROTOCOL** -300
- #define UPNP_E_SUBSCRIBE_UNACCEPTED -301

  A subscription request was rejected from the remote side.
- #define UPNP_E_UNSUBSCRIBE_UNACCEPTED -302

  An unsubscribe request was rejected from the remote side.
- #define UPNP_E_NOTIFY_UNACCEPTED -303

  The remote host did not accept the notify sent from the local device.
- #define UPNP_E_INVALID_ARGUMENT -501

  One or more of the parameters passed to a function is invalid.
- #define UPNP_E_FILE_NOT_FOUND -502

  The filename passed to one of the device registration functions was not found or was not accessible.
- #define UPNP_E_FILE_READ_ERROR -503

  An error happened while reading a file.
- #define UPNP_E_EXT_NOT_XML -504

  *The file name of the description document passed to **UpnpRegisterRootDevice2** does not end in ".xml".*
- #define **UPNP_E_NO_WEB_SERVER** -505
- #define **UPNP_E_OUTOF_BOUNDS** -506
- #define UPNP_E_NOT_FOUND -507

  The response to a SOAP request did not contain the required XML constructs.
- #define UPNP_E_INTERNAL_ERROR -911

  Generic error code for internal conditions not covered by other error codes.
- #define **UPNP_SOAP_E_INVALID_ACTION** 401
- #define **UPNP_SOAP_E_INVALID_ARGS** 402
- #define **UPNP_SOAP_E_OUT_OF_SYNC** 403
- #define **UPNP_SOAP_E_INVALID_VAR** 404
- #define **UPNP_SOAP_E_ACTION_FAILED** 501

7.1.1 Detailed Description

7.1.2 Macro Definition Documentation

7.1.2.1 IN6_IS_ADDR_GLOBAL

```
#define IN6_IS_ADDR_GLOBAL(
              a )
```

Value:

```
((((__const uint32_t *)(a))[0] & \
         htonl((uint32_t)0x70000000)) == \
        htonl((uint32_t)0x20000000))
```

7.1.2.2 IN6_IS_ADDR_ULA

```
#define IN6_IS_ADDR_ULA(
              a )
```

Value:

```
((((__const uint32_t *)(a))[0] & \
         htonl((uint32_t)0xfe000000)) == \
        htonl((uint32_t)0xfc000000))
```

7.1.2.3 UPNP_E_ALREADY_REGISTERED

```
#define UPNP_E_ALREADY_REGISTERED -120
```

A client or a device is already registered.

The SDK currently has a limit of one registered client and one registered device per process.

7.1.2.4 UPNP_E_BAD_HTTPMSG

```
#define UPNP_E_BAD_HTTPMSG -119
```

The HTTP message contains invalid message headers.

The error always refers to the HTTP message header received from the remote host. The main areas where this occurs are in SOAP control messages (e.g. **UpnpSendAction**), GENA subscription message (e.g. **Upnp↩ Subscribe**), GENA event notifications (e.g. **UpnpNotify**), and HTTP transfers (e.g. **UpnpDownloadXmlDoc**).

7.1.2.5 UPNP_E_BAD_RESPONSE

```
#define UPNP_E_BAD_RESPONSE -113
```

The response received from the remote side of a connection is not correct for the protocol.

This applies to the GENA, SOAP, and HTTP protocols.

7.1.2.6 UPNP_E_CANCELED

```
#define UPNP_E_CANCELED -210
```

The operation was canceled.

This error can be returned by any function that allows for external cancelation.

7.1.2.7 UPNP_E_EXT_NOT_XML

```
#define UPNP_E_EXT_NOT_XML -504
```

The file name of the description document passed to **UpnpRegisterRootDevice2** does not end in ".xml".

7.1.2.8 UPNP_E_FILE_NOT_FOUND

```
#define UPNP_E_FILE_NOT_FOUND -502
```

The filename passed to one of the device registration functions was not found or was not accessible.

7.1.2.9 UPNP_E_FILE_READ_ERROR

```
#define UPNP_E_FILE_READ_ERROR -503
```

An error happened while reading a file.

7.1.2.10 UPNP_E_FINISH

```
#define UPNP_E_FINISH -116
```

UpnpInit2 has not been called, or **UpnpFinish** has already been called.

None of the API functions operate until **UpnpInit2** successfully completes.

7.1.2.11 UPNP_E_INIT

```
#define UPNP_E_INIT -105
```

The SDK has already been initialized.

The SDK needs to be initialied only once per process. Any additional initialization attempts simply return this error with no other ill effects.

7.1.2.12 UPNP_E_INIT_FAILED

```
#define UPNP_E_INIT_FAILED -117
```

UpnpInit2 cannot complete.

The typical reason is failure to allocate sufficient resources.

7.1.2.13 UPNP_E_INTERNAL_ERROR

```
#define UPNP_E_INTERNAL_ERROR -911
```

Generic error code for internal conditions not covered by other error codes.

7.1.2.14 UPNP_E_INVALID_ACTION

```
#define UPNP_E_INVALID_ACTION -115
```

The SOAP action message is invalid.

This can be because the DOM document passed to the function was malformed or the action message is not correct for the given action.

7.1.2.15 UPNP_E_INVALID_ARGUMENT

```
#define UPNP_E_INVALID_ARGUMENT -501
```

One or more of the parameters passed to a function is invalid.

Refer to the individual function descriptions for the acceptable ranges for parameters.

7.1.2.16 UPNP_E_INVALID_DESC

```
#define UPNP_E_INVALID_DESC -107
```

The description document passed to **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2 UpnpRegister↩ RootDevice3** or **UpnpRegisterRootDevice4** is invalid.

7.1.2.17 UPNP_E_INVALID_HANDLE

```
#define UPNP_E_INVALID_HANDLE -100
```

The handle passed to a function is not a recognized as a valid handle.

7.1.2.18 UPNP_E_INVALID_INTERFACE

```
#define UPNP_E_INVALID_INTERFACE -121
```

The interface provided to **UpnpInit2** is unknown or does not have a valid IPv4 or IPv6 address configured.

7.1.2.19 UPNP_E_INVALID_PARAM

```
#define UPNP_E_INVALID_PARAM -101
```

One or more of the parameters passed to the function is not valid.

Refer to the documentation for each function for more information on the valid ranges of the parameters.

7.1.2.20 UPNP_E_INVALID_SERVICE

```
#define UPNP_E_INVALID_SERVICE -111
```

The device ID/service ID pair does not refer to a valid service.

Returned only by **UpnpNotify**, **UpnpNotifyExt**, **UpnpAcceptSubscription**, and **UpnpAcceptSubscriptionExt**.

7.1.2.21 UPNP_E_INVALID_URL

```
#define UPNP_E_INVALID_URL -108
```

An URL passed into the function is invalid.

The actual cause is function specific, but in general, the URL itself might be malformed (e.g. have invalid characters in it) or the host might be unreachable.

7.1.2.22 UPNP_E_LISTEN

```
#define UPNP_E_LISTEN -206
```

The SDK had a problem setting the socket to listen for incoming connections.

This error only happens during initialization (i.e. **UpnpInit2**).

7.1.2.23 UPNP_E_NETWORK_ERROR

```
#define UPNP_E_NETWORK_ERROR -200
```

A network error occurred.

It is the generic error code for network problems that are not covered under one of the more specific error codes. The typical meaning is the SDK failed to read the local IP address or had problems configuring one of the sockets.

7.1.2.24 UPNP_E_NOT_FOUND

```
#define UPNP_E_NOT_FOUND -507
```

The response to a SOAP request did not contain the required XML constructs.

7.1.2.25 UPNP_E_NOTIFY_UNACCEPTED

```
#define UPNP_E_NOTIFY_UNACCEPTED -303
```

The remote host did not accept the notify sent from the local device.

7.1.2.26 UPNP_E_OUTOF_HANDLE

```
#define UPNP_E_OUTOF_HANDLE -102
```

The SDK does not have any more space for additional handles.

The SDK allocates space for only a few handles in order to conserve memory.

7.1.2.27 UPNP_E_OUTOF_MEMORY

```
#define UPNP_E_OUTOF_MEMORY -104
```

Not enough resources are currently available to complete the operation.

Most operations require some free memory in order to complete their work.

7.1.2.28 UPNP_E_OUTOF_SOCKET

```
#define UPNP_E_OUTOF_SOCKET -205
```

The SDK cannot create any more sockets.

This occurs in any function that makes network connections, such as discovery (e.g. **UpnpSearchAsync** or **Upnp↩ SendAdvertisement**), control (e.g. **UpnpSendAction**), eventing (e.g. **UpnpNotify**), and HTTP functions (e.g. **UpnpDownloadXmlDoc**).

7.1.2.29 UPNP_E_SOCKET_BIND

```
#define UPNP_E_SOCKET_BIND -203
```

The SDK had a problem binding a socket to a network interface.

This occurs in any function that makes network connections, such as discovery (e.g. **UpnpSearchAsync** or **Upnp↩ SendAdvertisement**), control (e.g. **UpnpSendAction**), eventing (e.g. **UpnpNotify**), and HTTP functions (e.g. **UpnpDownloadXmlDoc**).

7.1.2.30 UPNP_E_SOCKET_CONNECT

```
#define UPNP_E_SOCKET_CONNECT -204
```

The SDK had a problem connecting to a remote host.

This occurs in any function that makes network connections, such as discovery (e.g. **UpnpSearchAsync** or **Upnp↩ SendAdvertisement**), control (e.g. **UpnpSendAction**), eventing (e.g. **UpnpNotify**), and HTTP functions (e.g. **UpnpDownloadXmlDoc**).

7.1.2.31 UPNP_E_SOCKET_ERROR

```
#define UPNP_E_SOCKET_ERROR -208
```

Generic socket error code for conditions not covered by other error codes.

This error can be returned by any function that performs network operations.

7.1.2.32 UPNP_E_SOCKET_READ

```
#define UPNP_E_SOCKET_READ -202
```

An error happened while reading from a socket.

This occurs in any function that makes network connections, such as discovery (e.g. **UpnpSearchAsync** or **UpnpSendAdvertisement**), control (e.g. **UpnpSendAction**), eventing (e.g. **UpnpNotify**), and HTTP functions (e.g. **UpnpDownloadXmlDoc**).

7.1.2.33 UPNP_E_SOCKET_WRITE

```
#define UPNP_E_SOCKET_WRITE -201
```

An error happened while writing to a socket.

This occurs in any function that makes network connections, such as discovery (e.g. **UpnpSearchAsync** or **UpnpSendAdvertisement**), control (e.g. **UpnpSendAction**), eventing (e.g. **UpnpNotify**), and HTTP functions (e.g. **UpnpDownloadXmlDoc**).

7.1.2.34 UPNP_E_SUBSCRIBE_UNACCEPTED

```
#define UPNP_E_SUBSCRIBE_UNACCEPTED -301
```

A subscription request was rejected from the remote side.

7.1.2.35 UPNP_E_SUCCESS

```
#define UPNP_E_SUCCESS 0
```

The operation completed successfully.

For asynchronous functions, this only means that the packet generated by the operation was successfully transmitted on the network. The result of the entire operation comes as part of the callback for that operation.

7.1.2.36 UPNP_E_TIMEDOUT

```
#define UPNP_E_TIMEDOUT -207
```

Too much time elapsed before the required number of bytes were sent or received over a socket.

This error can be returned by any function that performs network operations.

7.1.2.37 UPNP_E_UNSUBSCRIBE_UNACCEPTED

```
#define UPNP_E_UNSUBSCRIBE_UNACCEPTED -302
```

An unsubscribe request was rejected from the remote side.

7.1.2.38 UPNP_E_URL_TOO_BIG

```
#define UPNP_E_URL_TOO_BIG -118
```

The URL passed into a function is too long.

The SDK limits URLs to 180 characters in length.

7.1.3 Typedef Documentation

7.1.3.1 Upnp_SID

```
typedef char Upnp_SID[44]
```

Holds the subscription identifier for a subscription between a client and a device.

The SID is a string representation of a globally unique id (GUID) and should not be modified.

7.1.3.2 UpnpClient_Handle

```
typedef int UpnpClient_Handle
```

Returned when a control point application registers with **UpnpRegisterClient**.

Client handles can only be used with functions that operate with a client handle.

7.1.3.3 UpnpDevice_Handle

```
typedef int UpnpDevice_Handle
```

Returned when a device application registers with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **UpnpRegisterRootDevice3** or **UpnpRegisterRootDevice4**.

Device handles can only be used with functions that operate with a device handle.

7.1.3.4 UpnpWebFileHandle

```
typedef void* UpnpWebFileHandle
```

The type of handle returned by the web server for open requests.

7.1.3.5 VDCallback_Close

```
typedef int(* VDCallback_Close) ( UpnpWebFileHandle fileHnd, const void *cookie, const void
*request_cookie)
```

Close callback function prototype.

Parameters

in	*fileHnd*	The handle of the file to close.
in	*cookie*	The cookie associated with this VirtualDir
in	*request_cookie*	The cookie associated with this request

7.1.3.6 VDCallback_GetInfo

```
typedef int(* VDCallback_GetInfo) ( const char *filename, UpnpFileInfo *info, const void *cookie,
const void **request_cookie)
```

Get-info callback function prototype.

Parameters

in	*filename*	The name of the file to query.
out	*info*	Pointer to a structure to store the information on the file.
in	*cookie*	The cookie associated with this VirtualDir
out	*request_cookie*	The cookie associated with this request

7.1.3.7 VDCallback_Open

```
typedef UpnpWebFileHandle(* VDCallback_Open) ( const char *filename, enum UpnpOpenFileMode
Mode, const void *cookie, const void *request_cookie)
```

Open callback function prototype.

Parameters

in	*filename*	The name of the file to open.
in	*Mode*	The mode in which to open the file. Valid values are UPNP_READ or UPNP_WRITE.
in	*cookie*	The cookie associated with this VirtualDir
in	*request_cookie*	The cookie associated with this request

7.1.3.8 VDCallback_Read

```
typedef int(* VDCallback_Read) ( UpnpWebFileHandle fileHnd, char *buf, size_t buflen, const
void *cookie, const void *request_cookie)
```

Read callback function prototype.

Parameters

in	*fileHnd*	The handle of the file to read.
out	*buf*	The buffer in which to place the data.
in	*buflen*	The size of the buffer (i.e. the number of bytes to read).
in	*cookie*	The cookie associated with this VirtualDir
in	*request_cookie*	The cookie associated with this request

7.1.3.9 VDCallback_Seek

```
typedef int(* VDCallback_Seek) ( UpnpWebFileHandle fileHnd, off_t offset, int origin, const
void *cookie, const void *request_cookie)
```

Seek callback function prototype.

Parameters

in	*fileHnd*	The handle of the file to move the file pointer.
in	*offset*	The number of bytes to move in the file. Positive values move foward and negative values move backward. Note that this must be positive if the **origin** is SEEK_SET.
in	*origin*	The position to move relative to. It can be SEEK_CUR to move relative to the current position, SEEK_END to move relative to the end of the file, or SEEK_SET to specify an absolute offset.
in	*cookie*	The cookie associated with this VirtualDir
in	*request_cookie*	The cookie associated with this request

7.1.3.10 VDCallback_Write

```
typedef int(* VDCallback_Write) ( UpnpWebFileHandle fileHnd, char *buf, size_t buflen, const
void *cookie, const void *request_cookie)
```

Write callback function prototype.

Parameters

in	*fileHnd*	The handle of the file to write.
in	*buf*	The buffer with the bytes to write.
in	*buflen*	The number of bytes to write.
in	*cookie*	The cookie associated with this VirtualDir
in	*request_cookie*	The cookie associated with this request

7.1.4 Enumeration Type Documentation

7.1.4.1 Upnp_DescType_e

```
enum Upnp_DescType_e
```

Specifies the type of description in **UpnpRegisterRootDevice2**.

These values control how **UpnpRegisterRootDevice2** interprets the **description** parameter.

Enumerator

UPNPREG_URL_DESC	The description is the URL to the description document.
UPNPREG_FILENAME_DESC	The description is a file name on the local file system containing the description of the device.
UPNPREG_BUF_DESC	The description is a pointer to a character array containing the XML description document.

7.1.4.2 Upnp_HttpMethod_e

```
enum Upnp_HttpMethod_e
```

Different HTTP methods.

7.1.4.3 Upnp_SType_e

```
enum Upnp_SType_e
```

Represents the different types of searches that can be performed using the SDK for UPnP Devices API.

By specifying these different values to **UpnpSearchAsync**, the control point application can control the scope of the search from all devices to specific devices or services.

Enumerator

UPNP_S_ALL	Search for all devices and services on the network.
UPNP_S_ROOT	Search for all root devices on the network.
UPNP_S_DEVICE	Search for a particular device type or a particular device instance.
UPNP_S_SERVICE	Search for a particular service type, possibly on a particular device type or device instance.

7.1.5 Function Documentation

7.1.5.1 AutoAdvertise()

```
void AutoAdvertise (
             void * input )
```

This function is a timer thread scheduled by UpnpSendAdvertisement to the send advetisement again.

Parameters

in	*input*	Information provided to the thread.

References free_advertise_arg(), and UpnpSendAdvertisement().

Referenced by UpnpSendAdvertisementLowPower().

7.1.5.2 free_action_arg()

```
static void free_action_arg (
             job_arg * arg )  [static]
```

Free memory associated with an action job's argument.

References ixmlDocument_free().

Referenced by UpnpSendActionAsync(), UpnpSendActionExAsync(), and UpnpThreadDistribution().

7.1.5.3 free_advertise_arg()

```
static void free_advertise_arg (
            job_arg * arg )  [static]
```

Free memory associated with advertise job's argument.

Referenced by AutoAdvertise(), and UpnpSendAdvertisementLowPower().

7.1.5.4 FreeHandle()

```
static int FreeHandle (
            int Upnp_Handle )  [static]
```

Free handle.

Returns

UPNP_E_SUCCESS if successful or UPNP_E_INVALID_HANDLE if not

Parameters

in	*Upnp_Handle*	Handle index.

References HandleTable, UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), UpnpUnRegisterClient(), and UpnpUnRegisterRootDeviceLowPower().

7.1.5.5 GetCallBackFn()

```
Upnp_FunPtr GetCallBackFn (
            UpnpClient_Handle Hnd )
```

Get callback function ptr from a handle.

Returns

Upnp_FunPtr

References HandleTable.

7.1.5.6 GetClientHandleInfo()

```
Upnp_Handle_Type GetClientHandleInfo (
            int * client_handle_out,
            struct Handle_Info ** HndInfo )
```

Get client handle info.

Note

The logic around the use of this function should be revised.

Returns

HND_CLIENT, HND_INVALID

Parameters

in	*client_handle_out*	client handle pointer (key for the client handle structure).
out	*HndInfo*	Client handle structure passed by this function.

References GetHandleInfo().

Referenced by gena_process_notification_event(), ssdp_handle_ctrlpt_msg(), and UpnpFinish().

7.1.5.7 GetDescDocumentAndURL()

```
static int GetDescDocumentAndURL (
            Upnp_DescType descriptionType,
            char * description,
            int config_baseURL,
            int AddressFamily,
            IXML_Document ** xmlDoc,
            char descURL[LINE_SIZE] )  [static]
```

Fills the sockadr_in with miniserver information.

References UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UPNP_E_URL_TOO_BIG, UpnpDownloadXmlDoc(), and UPNPREG_URL_DESC.

Referenced by UpnpRegisterRootDevice2().

7.1.5.8 GetDeviceHandleInfo()

```
Upnp_Handle_Type GetDeviceHandleInfo (
            UpnpDevice_Handle start,
            int AddressFamily,
            int * device_handle_out,
            struct Handle_Info ** HndInfo )
```

Retrieves the device handle and information of the first device of the address family specified. The search begins at the 'start' index, which should be 0 for the first call, then the last successful value returned. This allows listing all entries for the address family.

Returns

HND_DEVICE or HND_INVALID

Parameters

in	*start*	place to start the search (i.e. last value returned).
in	*AddressFamily*	Address family.
out	*device_handle_out*	Device handle pointer.
out	*HndInfo*	Device handle structure passed by this function.

References GetHandleInfo(), UpnpSdkDeviceRegisteredV4, and UpnpSdkDeviceregisteredV6.

Referenced by ssdp_handle_device_request(), and UpnpFinish().

7.1.5.9 GetDeviceHandleInfoForPath()

```
Upnp_Handle_Type GetDeviceHandleInfoForPath (
            const char * path,
            int AddressFamily,
            int * device_handle_out,
            struct Handle_Info ** HndInfo,
            service_info ** serv_info )
```

Retrieves the device handle and information of the first device of the address family specified, with a service having a controlURL or eventSubURL matching the path.

Returns

HND_DEVICE or HND_INVALID

Parameters

	path	The Uri path.
in	*AddressFamily*	Address family.
out	*device_handle_out*	Device handle pointer.
out	*HndInfo*	Device handle structure passed by this function.
out	*serv_info*	Service info for found path.

References GetHandleInfo(), UpnpSdkDeviceRegisteredV4, and UpnpSdkDeviceregisteredV6.

Referenced by gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_re and get_dev_service().

7.1.5.10 GetFreeHandle()

```
static int GetFreeHandle ( )  [static]
```

Get a free handle.

Returns

On success, an integer greater than zero or UPNP_E_OUTOF_HANDLE on failure.

References HandleTable, and UPNP_E_OUTOF_HANDLE.

Referenced by UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

7.1.5.11 GetHandleInfo()

```
Upnp_Handle_Type GetHandleInfo (
            int Hnd,
            struct Handle_Info ** HndInfo )
```

Get handle information.

Returns

HND_DEVICE, UPNP_E_INVALID_HANDLE

Parameters

Hnd	handle pointer (key for the client handle structure).
HndInfo	handle structure passed by this function.

References HandleTable, Handle_Info::HType, and UpnpPrintf().

Referenced by AdvertiseAndReply(), gena_process_notification_event(), GenaAutoRenewSubscription(), genaNotifyThread(), genaRenewSubscription(), genaSubscribe(), genaUnregisterClient(), genaUnregisterDevice(), genaUnSubscribe(), GetClientHandleInfo(), GetDeviceHandleInfo(), GetDeviceHandleInfoForPath(), SearchByTarget(), ssdp_handle_ctrlpt_msg(), UpnpAcceptSubscription(), UpnpAcceptSubscriptionExt(), UpnpGetServiceVarStatus(), UpnpGetServiceVarStatusAsync(), UpnpNotify(), UpnpNotifyExt(), UpnpRenewSubscription(), UpnpRenewSubscriptionAsync(), UpnpSearchAsync(), UpnpSendAction(), UpnpSendActionAsync(), UpnpSendActionEx(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSetContentLength(), UpnpSetMaxSubscriptions(), UpnpSetMaxSubscriptionTimeOut(), UpnpSubscribe(), UpnpSubscribeAsync(), UpnpUnRegisterClient(), UpnpUnRegisterRootDeviceLowPower(), UpnpUnSubscribe(), and UpnpUnSubscribeAsync().

7.1.5.12 PrintHandleInfo()

```
int PrintHandleInfo (
            UpnpClient_Handle Hnd )
```

Print handle info.

Returns

UPNP_E_SUCCESS if successful, otherwise returns appropriate error.

Parameters

in	*Hnd*	Handle index.

References Handle_Info::DescURL, HandleTable, Handle_Info::HType, UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, and UpnpPrintf().

7.1.5.13 PrintThreadPoolStats()

```
void PrintThreadPoolStats (
            ThreadPool * tp,
            const char * DbgFileName,
            int DbgLineNo,
            const char * msg )
```

Prints thread pool statistics.

Parameters

in	*tp*	The thread pool.
in	*DbgFileName*	The file name that called this function, use the macro **FILE**.
in	*DbgLineNo*	The line number that the function was called, use the macro **LINE**.
in	*msg*	The message.

References ThreadPoolGetStats(), and UpnpPrintf().

Referenced by UpnpFinish().

7.1.5.14 UpnpAcceptSubscription()

```
int UpnpAcceptSubscription (
             UpnpDevice_Handle Hnd,
             const char * DevID,
             const char * ServID,
             const char ** VarName,
             const char ** NewVal,
             int cVariables,
             const Upnp_SID SubsId )
```

Accepts a subscription request and sends out the current state of the eventable variables for a service.

The device application should call this function when it receives a UPNP_EVENT_SUBSCRIPTION_REQUEST callback.

This function is synchronous and generates no callbacks.

This function can be called during the execution of a callback function.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid device handle.
- UPNP_E_INVALID_SERVICE: The **DevId/ServId pair** refers to an invalid service.
- UPNP_E_INVALID_SID: The specified subscription ID is not valid.
- UPNP_E_INVALID_PARAM: Either **VarName**, **NewVal**, **DevID**, or **ServID** is not a valid pointer or **cVariables** is less than zero.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the device.
in	*DevID*	The device ID of the subdevice of the service generating the event.
in	*ServID*	The unique service identifier of the service generating the event.
in	*VarName*	Pointer to an array of event variables.
in	*NewVal*	Pointer to an array of values for the event variables.
in	*cVariables*	The number of event variables in **VarName**.
in	*SubsId*	The subscription ID of the newly registered control point.

References genaInitNotify(), GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UpnpPrintf(), and UpnpSdkInit.

Referenced by TvDeviceHandleSubscriptionRequest().

7.1.5.15 UpnpAcceptSubscriptionExt()

```
int UpnpAcceptSubscriptionExt (
            UpnpDevice_Handle Hnd,
            const char * DevID,
            const char * ServID,
            IXML_Document * PropSet,
            const Upnp_SID SubsId )
```

Similar to **UpnpAcceptSubscription()** except that it takes a DOM document for the variables to event rather than an array of strings.

This function is sychronous and generates no callbacks.

This function can be called during the execution of a callback function.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid device handle.
- `UPNP_E_INVALID_SERVICE`: The **DevId/ServId pair** refers to an invalid service.
- `UPNP_E_INVALID_SID`: The specified subscription ID is not valid.
- `UPNP_E_INVALID_PARAM`: Either **VarName**, **NewVal**, **DevID**, **ServID**, or **PropSet** is not a valid pointer.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the device.
in	*DevID*	The device ID of the subdevice of the service generating the event.
in	*ServID*	The unique service identifier of the service generating the event.
in	*PropSet*	The DOM document for the property set. Property set documents must conform to the XML schema defined in section 4.3 of the Universal Plug and Play Device Architecture specification.
in	*SubsId*	The subscription ID of the newly registered control point.

References genaInitNotifyExt(), GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UpnpPrintf(), and UpnpSdkInit.

Referenced by TvDeviceHandleSubscriptionRequest().

7.1.5.16 UpnpAddVirtualDir()

```
int UpnpAddVirtualDir (
            const char * dirName,
            const void * cookie,
            const void ** oldcookie )
```

Adds a virtual directory mapping.

All webserver requests containing the given directory are read using functions contained in a **VirtualDirCallbacks** structure registered via **UpnpSetVirtualDirCallbacks**.

Note

This function is not available when the web server is not compiled into the UPnP Library.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **dirName** is not valid.

Parameters

`in`	*dirName*	The name of the new directory mapping to add.
`in`	*cookie*	The cookie to associated with this virtual directory
`out`	*oldcookie*	The cookie previously associated, if mapping is already present

References pVirtualDirList, UPNP_E_FINISH, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, and UpnpSdkInit.

7.1.5.17 UpnpCancelHttpGet()

```
int UpnpCancelHttpGet (
            void * handle )
```

Set the cancel flag of the **handle** parameter.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: **handle** is not a valid pointer.

Parameters

`in`	*handle*	The handle of the connection created by the call to **UpnpOpenHttpGet**.

7.1.5.18 UpnpCloseHttpConnection()

```
int UpnpCloseHttpConnection (
            void * handle )
```

Closes the connection created with **UpnpOpenHttpConnection** and frees any memory associated with the connection.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: **handle**, or is not a valid pointer.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*handle*	The handle of the connection to close, created by the call to **UpnpOpenHttpPost**.

References http_CloseHttpConnection().

Referenced by UpnpCloseHttpGet().

7.1.5.19 UpnpCloseHttpGet()

```
int UpnpCloseHttpGet (
            void * handle )
```

Closes the connection and frees memory that was allocated for the **handle** parameter.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: **handle** is not a valid pointer.

Parameters

in	*handle*	The handle of the connection created by the call to **UpnpOpenHttpGet**.

References UpnpCloseHttpConnection().

7.1.5.20 UpnpCloseHttpPost()

```
int UpnpCloseHttpPost (
            void * handle,
            int * httpStatus,
            int timeout )
```

Sends and receives any pending data, closes the connection with the server, and frees memory allocated during the **UpnpOpenHttpPost** call.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: Either **handle**, or **httpStatus** is not a valid pointer.
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.

Parameters

in	*handle*	The handle of the connection to close, created by the call to **UpnpOpenHttpPost**.
in,out	*httpStatus*	A pointer to a buffer to store the final status of the connection.
in	*timeout*	A time out value sent with the request during which a response is expected from the server, failing which, an error is reported. If value is negative, timeout is infinite.

References http_CloseHttpConnection(), http_EndHttpRequest(), http_GetHttpResponse(), and UPNP_E_SUCCESS.

7.1.5.21 UpnpComputeIpv6PrefixLength()

```
static unsigned UpnpComputeIpv6PrefixLength (
            struct sockaddr_in6 * Netmask )  [static]
```

Computes prefix length from IPv6 netmask.

Returns

The IPv6 prefix length.

Referenced by UpnpGetIfInfo().

7.1.5.22 UpnpDownloadUrlItem()

```
int UpnpDownloadUrlItem (
            const char * url,
            char ** outBuf,
            char * contentType )
```

Downloads a file specified in a URL.

The SDK allocates the memory for **outBuf** and the application is responsible for freeing this memory. Note that the item is passed as a single buffer. Large items should not be transferred using this function.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: Either **url**, **outBuf** or **contentType** is not a valid pointer.
- UPNP_E_INVALID_URL: The **url** is not a valid URL.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to download this file.
- UPNP_E_NETWORK_ERROR: A network error occurred.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket.
- UPNP_E_SOCKET_BIND: An error occurred binding a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting a socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.

Parameters

in	*url*	URL of an item to download.
out	*outBuf*	Buffer to store the downloaded item.
out	*contentType*	HTTP header value content type if present. It should be at least LINE_SIZE bytes in size.

References UPNP_E_INVALID_PARAM, and UPNP_E_INVALID_URL.

Referenced by UpnpDownloadXmlDoc().

7.1.5.23 UpnpDownloadXmlDoc()

```
int UpnpDownloadXmlDoc (
            const char * url,
            IXML_Document ** xmlDoc )
```

Downloads an XML document specified in a URL.

The SDK parses the document and returns it in the form of a DOM document. The application is responsible for freeing the DOM document.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: Either **url** or **xmlDoc** is not a valid pointer.
- UPNP_E_INVALID_DESC: The XML document was not found or it does not contain a valid XML description.
- UPNP_E_INVALID_URL: The **url** is not a valid URL.
- UPNP_E_OUTOF_MEMORY: There are insufficient resources to download the XML document.
- UPNP_E_NETWORK_ERROR: A network error occurred.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket.
- UPNP_E_SOCKET_BIND: An error occurred binding a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting the socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.

Parameters

in	*url*	URL of the XML document.
out	*xmlDoc*	A pointer in which to store the XML document.

References ixmlFreeDOMString(), ixmlParseBufferEx(), ixmlPrintNode(), UPNP_E_INVALID_DESC, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpDownloadUrlItem(), and UpnpPrintf().

Referenced by GetDescDocumentAndURL(), TvDeviceStateTableInit(), UpnpRegisterRootDevice(), and UpnpRegisterRootDevice4().

7.1.5.24 UpnpEnableWebserver()

```
int UpnpEnableWebserver (
            int enable )
```

Enables or disables the webserver.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_OUTOF_MEMORY`: The web server could not be started due to an out-of-memory condition.
- `UPNP_E_NO_WEB_SERVER`: The internal web server has been compiled out so it can't be enabled or disabled.

Parameters

in	*enable*	`1` to enable, `0` to disable.

References bWebServerState, SetHTTPGetCallback(), UPNP_E_FINISH, UPNP_E_SUCCESS, UpnpSdkInit, web_server_callback(), web_server_destroy(), and web_server_init().

Referenced by UpnpInitStartServers().

7.1.5.25 UpnpEndHttpRequest()

```
int UpnpEndHttpRequest (
            void * handle,
            int timeout )
```

Indicates the end of a HTTP request previously made by **UpnpMakeHttpRequest**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: **handle** is not a valid pointer.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_SOCKET_ERROR`: Error occured allocating a socket and resources or an error occurred binding a socket.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*handle*	The handle to the connection.
in	*timeout*	The time out value sent with the request during which a response is expected from the receiver, failing which, an error is reported. If value is negative, timeout is infinite.

References http_EndHttpRequest().

Referenced by UpnpOpenHttpGet(), and UpnpOpenHttpGetProxy().

7.1.5.26 UpnpFinish()

```
int UpnpFinish (
            void  )
```

Initializes the OpenSSL library, and the OpenSSL context for use with pupnp.

Note

This method is only enabled if pupnp is compiled with open-ssl support.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INIT`: The SDK is already initialized.
- `UPNP_E_INIT_FAILED`: The SDK initialization failed for an unknown reason.

Terminates the Linux SDK for UPnP Devices.

- Checks for pending jobs and threads
- Unregisters either the client or device
- Shuts down the Timer Thread
- Stops the Mini Server
- Uninitializes the Thread Pool
- For Win32 cleans up Winsock Interface
- Cleans up mutex objects

This function must be the last API function called. It should be called only once. Subsequent calls to this API return a `UPNP_E_FINISH` error code.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_FINISH`: The SDK is already terminated or it is not initialized.

References device_handle, GetClientHandleInfo(), GetDeviceHandleInfo(), GlobalClientSubscribeMutex, GlobalHndRWLock, gMiniServerThreadPool, gRecvThreadPool, gSendThreadPool, gTimerThread, gUUIDMutex, PrintThreadPoolStats(), StopMiniServer(), ThreadPoolShutdown(), TimerThreadShutdown(), UPNP_E_FINISH, UPNP_E_SUCCESS, UpnpCloseLog(), UpnpPrintf(), UpnpRemoveAllVirtualDirs(), UpnpSdkInit, UpnpUnRegisterClient(), UpnpUnRegisterRootDevice(), and web_server_destroy().

Referenced by TvCtrlPointStart(), TvDeviceStart(), TvDeviceStop(), UpnpInit2(), UpnpInitPreamble(), UpnpInitStartServers(), and UpnpInitThreadPools().

7.1.5.27 UpnpGetHttpResponse()

```
int UpnpGetHttpResponse (
            void * handle,
            UpnpString * headers,
            char ** contentType,
            int * contentLength,
            int * httpStatus,
            int timeout )
```

Gets the response from the server using a connection previously created by **UpnpOpenHttpConnection**.

Note

Memory for **contentType** is only valid until the next call to the HTTP API for the same connection.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, is not a valid pointer.
- `UPNP_E_INVALID_URL`: The **url** is not a valid URL.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_NETWORK_ERROR`: A network error occurred.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.
- `UPNP_E_BAD_RESPONSE`: A bad response was received from the remote server.

Parameters

in	*handle*	The handle of the connection created by the call to **UpnpOpenHttpConnection**.
in	*headers*	Headers sent by the server for the response. If NULL then the headers are not copied.
out	*contentType*	A buffer to store the media type of the item.
out	*contentLength*	A pointer to store the length of the item.
out	*httpStatus*	The status returned on receiving a response message.
in	*timeout*	The time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is negative, timeout is infinite.

References http_GetHttpResponse().

Referenced by UpnpOpenHttpGet(), and UpnpOpenHttpGetProxy().

7.1.5.28 UpnpGetIfInfo()

```
int UpnpGetIfInfo (
```

```
            const char * IfName )
```

Retrieve interface information and keep it in global variables. If NULL, we'll find the first suitable interface for operation.

The interface must fulfill these requirements:

- Be UP.
- Not be LOOPBACK.
- Support MULTICAST.
- Have a valid IPv4 or IPv6 address.

We'll retrieve the following information from the interface:

- gIF_NAME -> Interface name (by input or found).
- gIF_IPV4 -> IPv4 address (if any).
- gIF_IPV6 -> IPv6 address (if any).
- gIF_IPV6_ULA_GUA -> ULA or GUA IPv6 address (if any)
- gIF_INDEX -> Interface index number.

Returns

UPNP_E_SUCCESS on success.

Parameters

in	*IfName*	Interface name (can be NULL).

References gIF_INDEX, gIF_IPV4, gIF_IPV4_NETMASK, gIF_IPV6, gIF_IPV6_PREFIX_LENGTH, gIF_IPV6_ULA_GUA, gIF_IPV6_ULA_GUA_PREFIX_LENGTH, gIF_NAME, UPNP_E_INIT, UPNP_E_INVALID_INTERFACE, UPNP_E_OUTOF_MEMOR UPNP_E_SUCCESS, UpnpComputeIpv6PrefixLength(), and UpnpPrintf().

Referenced by UpnpInit2().

7.1.5.29 UpnpGetServerIp6Address()

```
char * UpnpGetServerIp6Address (
            void  )
```

Returns the IPv6 link-local listening ip address.

If NULL is used as the interface in **UpnpInit2**, then this function can be used to retrieve the actual interface address on which device is running.

Returns

- On success: The IPv6 link-local address (LLA) on which an internal server is listening for UPnP related requests.
- On error: `NULL` is returned if **UpnpInit2** has not succeeded.

References gIF_IPV6, and UpnpSdkInit.

Referenced by TvCtrlPointStart(), and TvDeviceStart().

7.1.5.30 UpnpGetServerIpAddress()

```
char * UpnpGetServerIpAddress (
            void  )
```

Returns the local IPv4 listening ip address.

If `NULL` is used as the interface in **UpnpInit2**, then this function can be used to retrieve the actual interface address on which device is running.

Returns

- On success: The IPv4 address on which an internal server is listening for UPnP related requests.
- On error: `NULL` is returned if **UpnpInit2** has not succeeded.

References gIF_IPV4, and UpnpSdkInit.

Referenced by TvCtrlPointStart(), and TvDeviceStart().

7.1.5.31 UpnpGetServerPort()

```
unsigned short UpnpGetServerPort (
            void  )
```

Returns the internal server IPv4 UPnP listening port.

If '0' is used as the port number in **UpnpInit2**, then this function can be used to retrieve the actual port allocated to the SDK.

Returns

- On success: The port on which an internal server is listening for IPv4 UPnP related requests.
- On error: 0 is returned if **UpnpInit2** has not succeeded.

References LOCAL_PORT_V4, and UpnpSdkInit.

Referenced by TvCtrlPointStart(), and TvDeviceStart().

7.1.5.32 UpnpGetServerPort6()

```
unsigned short UpnpGetServerPort6 (
            void  )
```

Returns the internal server IPv6 link-local (LLA) UPnP listening port.

If '0' is used as the port number in **UpnpInit2**, then this function can be used to retrieve the actual port allocated to the SDK.

Returns

- On success: The port on which an internal server is listening for IPv6 link-local (LLA) UPnP related requests.
- On error: 0 is returned if **UpnpInit2** has not succeeded.

References LOCAL_PORT_V6, and UpnpSdkInit.

Referenced by TvCtrlPointStart(), and TvDeviceStart().

7.1.5.33 UpnpGetServerUlaGuaIp6Address()

```
char * UpnpGetServerUlaGuaIp6Address (
            void  )
```

Returns the IPv6 unique-local or globally-unique listening ip address.

If `NULL` is used as the interface in **UpnpInit2**, then this function can be used to retrieve the actual interface address on which device is running.

Returns

- On success: The IPv6 unique-local or globally-unique address (ULA or GUA) on which an internal server is listening for UPnP related requests.
- On error: `NULL` is returned if **UpnpInit2** has not succeeded.

References gIF_IPV6_ULA_GUA, and UpnpSdkInit.

Referenced by TvCtrlPointStart(), and TvDeviceStart().

7.1.5.34 UpnpGetServerUlaGuaPort6()

```
unsigned short UpnpGetServerUlaGuaPort6 (
            void  )
```

Returns the internal server IPv6 ULA or GUA UPnP listening port.

If '0' is used as the port number in **UpnpInit2**, then this function can be used to retrieve the actual port allocated to the SDK.

Returns

- On success: The port on which an internal server is listening for IPv6 ULA or GUA UPnP related requests.
- On error: 0 is returned if **UpnpInit2** has not succeeded.

References LOCAL_PORT_V6_ULA_GUA, and UpnpSdkInit.

Referenced by TvCtrlPointStart(), and TvDeviceStart().

7.1.5.35 UpnpGetServiceVarStatus()

```
int UpnpGetServiceVarStatus (
            UpnpClient_Handle Hnd,
            const char * ActionURL,
            const char * VarName,
            DOMString * StVarVal )
```

Queries the state of a state variable of a service on another device.

Deprecated **The use of this function is deprecated by the UPnP Forum**.

This is a synchronous call.

A positive return value indicates a SOAP error code, whereas a negative return code indicates an SDK error code.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid control point handle.
- `UPNP_E_INVALID_URL`: **ActionUrl** is not a valid URL.
- `UPNP_E_INVALID_DESC`: The XML document was not found or it does not contain a valid XML description.
- `UPNP_E_INVALID_PARAM`: **StVarVal** is not a valid pointer or **VarName** or **ActionUrl** is `NULL`.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.
- `UPNP_SOAP_E_INVALID_VAR`: The given variable is invalid according to the device.

Parameters

`in`	*Hnd*	The handle of the control point.
`in`	*ActionURL*	The URL of the service.
`in`	*VarName*	The name of the variable to query.
`out`	*StVarVal*	The pointer to store the value for **VarName**. The SDK allocates this string and the caller needs to free it using **ixmlFreeDOMString**.

References GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UpnpPrintf(), and UpnpSdkInit.

7.1.5.36 UpnpGetServiceVarStatusAsync()

```
int UpnpGetServiceVarStatusAsync (
            UpnpClient_Handle Hnd,
            const char * ActionURL,
            const char * VarName,
            Upnp_FunPtr Fun,
            const void * Cookie )
```

Queries the state of a variable of a service, generating a callback when the operation is complete.

Deprecated **The use of this function is deprecated by the UPnP Forum.**

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid control point handle.
- `UPNP_E_INVALID_URL`: The **ActionUrl** is not a valid URL.
- `UPNP_E_INVALID_PARAM`: **VarName**, **Fun** or **ActionUrl** is not a valid pointer.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the control point.
in	*ActionURL*	The URL of the service.
in	*VarName*	The name of the variable to query.
in	*Fun*	Pointer to a callback function to be invoked when the operation is complete.
in	*Cookie*	Pointer to user data to pass to the callback function when invoked.

References GetHandleInfo(), gSendThreadPool, ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkInit, and UpnpThreadDistribution().

7.1.5.37 UpnpHttpGetProgress()

```
int UpnpHttpGetProgress (
            void * handle,
            size_t * length,
            size_t * total )
```

Retrieve progress information of a http-get transfer.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, **length** or **total** is not a valid pointer.

Parameters

in	*handle*	The token created by the call to **UpnpOpenHttpGet**.
out	*length*	The number of bytes received.
out	*total*	The content length.

7.1.5.38 UpnpInit2()

```
int UpnpInit2 (
            const char * IfName,
            unsigned short DestPort )
```

Initializes the Linux SDK for UPnP Devices (IPv4 or IPv6).

This function must be called before any other API function can be called. It should be called only once. Subsequent calls to this API return a UPNP_E_INIT error code.

Optionally, the application can specify an interface name (in the case of a multi-homed configuration) and a port number to use for all UPnP operations. Since a port number can be used only by one process, multiple processes using the SDK must specify different port numbers.

If unspecified, the SDK will use the first suitable interface and an arbitrary port.

This call is synchronous.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to initialize the SDK.
- UPNP_E_INIT: The SDK is already initialized.
- UPNP_E_INIT_FAILED: The SDK initialization failed for an unknown reason.
- UPNP_E_SOCKET_BIND: An error occurred binding a socket.
- UPNP_E_LISTEN: An error occurred listening to a socket.
- UPNP_E_OUTOF_SOCKET: An error ocurred creating a socket.
- UPNP_E_INTERNAL_ERROR: An internal error ocurred.
- UPNP_E_INVALID_INTERFACE: IfName is invalid or does not have a valid IPv4 or IPv6 addresss configured.

Parameters

IfName	The interface name to use by the UPnP SDK operations. Examples: "eth0", "xl0", "Local Area Connection", NULL to use the first suitable interface.
DestPort	Local Port to listen for incoming connections. NULL will pick an arbitrary free port.

References gSDKInitMutex, UPNP_E_INIT, UPNP_E_SUCCESS, UpnpFinish(), UpnpGetIfInfo(), UpnpInitPreamble(), UpnpInitStartServers(), UpnpPrintf(), and UpnpSdkInit.

Referenced by TvCtrlPointStart(), and TvDeviceStart().

7.1.5.39 UpnpInitMutexes()

```
static int UpnpInitMutexes (
            void ) [static]
```

Initializes the global mutexes used by the UPnP SDK.

Returns

UPNP_E_SUCCESS on success or UPNP_E_INIT_FAILED if a mutex could not be initialized.

References GlobalClientSubscribeMutex, GlobalHndRWLock, gUUIDMutex, UPNP_E_INIT_FAILED, and UPNP_E_SUCCESS.

Referenced by UpnpInitPreamble().

7.1.5.40 UpnpInitPreamble()

```
static int UpnpInitPreamble (
            void  )  [static]
```

Performs the initial steps in initializing the UPnP SDK.

- Winsock library is initialized for the process (Windows specific).
- The logging (for debug messages) is initialized.
- Mutexes, Handle table and thread pools are allocated and initialized.
- Callback functions for SOAP and GENA are set, if they're enabled.
- The SDK timer thread is initialized.

Returns

UPNP_E_SUCCESS on success.

References gSendThreadPool, gTimerThread, gUpnpSdkNLSuuid, HandleTable, SetGenaCallback(), SetSoapCallback(), soap_device_callback(), TimerThreadInit(), UPNP_E_INIT_FAILED, UPNP_E_SUCCESS, UpnpFinish(), UpnpInitLog(), UpnpInitMutexes(), UpnpInitThreadPools(), UpnpPrintf(), and WinsockInit().

Referenced by UpnpInit2().

7.1.5.41 UpnpInitStartServers()

```
static int UpnpInitStartServers (
            unsigned short DestPort )  [static]
```

Finishes initializing the UPnP SDK.

- The MiniServer is started, if enabled.
- The WebServer is started, if enabled.

Returns

UPNP_E_SUCCESS on success or UPNP_E_INIT_FAILED if a mutex could not be initialized.

Parameters

in	*DestPort*	Local Port to listen for incoming connections.

References gDocumentRootDir, gWebserverCorsString, LOCAL_PORT_V4, LOCAL_PORT_V6, LOCAL_PORT_V6_ULA_GUA, StartMiniServer(), UPNP_E_SUCCESS, UpnpEnableWebserver(), UpnpFinish(), and UpnpPrintf().

Referenced by UpnpInit2().

7.1.5.42 UpnpInitThreadPools()

```
static int UpnpInitThreadPools (
            void  )  [static]
```

Initializes the global threadm pools used by the UPnP SDK.

Returns

UPNP_E_SUCCESS on success or UPNP_E_INIT_FAILED if a mutex could not be initialized.

References gMiniServerThreadPool, gRecvThreadPool, gSendThreadPool, ThreadPoolInit(), TPAttrInit(), TPAttrSetIdleTime(), TPAttrSetJobsPerThread(), TPAttrSetMaxJobsTotal(), TPAttrSetMaxThreads(), TPAttrSetMinThreads(), TPAttrSetStackSize(), UPNP_E_INIT_FAILED, UPNP_E_SUCCESS, UpnpFinish(), and UpnpSdkInit.

Referenced by UpnpInitPreamble().

7.1.5.43 UpnpIsWebserverEnabled()

```
int UpnpIsWebserverEnabled (
            void  )
```

Returns `1` if the webserver is enabled, or `0` if it is not.

Checks if the webserver is enabled or disabled.

Returns

An integer representing one of the following:

- `1`: The webserver is enabled.
- `0`: The webserver is not enabled

1, if webserver is enabled or 0, if webserver is disabled.

References bWebServerState, and UpnpSdkInit.

7.1.5.44 UpnpMakeHttpRequest()

```
int UpnpMakeHttpRequest (
              Upnp_HttpMethod method,
              const char * url,
              void * handle,
              UpnpString * headers,
              const char * contentType,
              int contentLength,
              int timeout )
```

Makes a HTTP request using a connection previously created by **UpnpOpenHttpConnection**.

Note

Trying to make another request while a request is already being processed results in undefined behavior. It's up to the user to end a previous request by calling **UpnpEndHttpRequest**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **url**, **handle** or **contentType** is not a valid pointer.
- `UPNP_E_INVALID_URL`: The **url** is not a valid URL.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_SOCKET_ERROR`: Error occured allocating a socket and resources or an error occurred binding a socket.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*url*	The URL to use to make the request. The URL should use the same scheme used to create the connection, but the host can be different if the request is being proxied.
in	*handle*	The handle to the connection.
in	*headers*	Headers to be used for the request. Each header should be terminated by a CRLF as specified in the HTTP specification. If NULL then the default headers will be used.
in	*contentType*	The media type of content being sent. Can be NULL.
in	*contentLength*	The length of the content being sent, in bytes. Set to **UPNP_USING_CHUNKED** to use chunked encoding, or **UPNP_UNTIL_CLOSE** to avoid specifying the content length to the server. In this case the request is considered unfinished until the connection is closed.
in	*timeout*	The time out value sent with the request during which a response is expected from the receiver, failing which, an error is reported. If value is negative, timeout is infinite.

References http_MakeHttpRequest().

Referenced by UpnpOpenHttpGet(), and UpnpOpenHttpGetProxy().

7.1.5.45 UpnpNotify()

```
int UpnpNotify (
              UpnpDevice_Handle Hnd,
              const char * DevID,
              const char * ServID,
              const char ** VarName,
              const char ** NewVal,
              int cVariables )
```

Sends out an event change notification to all control points subscribed to a particular service.

This function is synchronous and generates no callbacks.

This function may be called during a callback function to send out a notification.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid device handle.
- `UPNP_E_INVALID_SERVICE`: The **DevId/ServId pair** refers to an invalid service.
- `UPNP_E_INVALID_PARAM`: Either **VarName**, **NewVal**, **DevID**, or **ServID** is not a valid pointer or **cVariables** is less than zero.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle to the device sending the event.
in	*DevID*	The device ID of the subdevice of the service generating the event.
in	*ServID*	The unique identifier of the service generating the event.
in	*VarName*	Pointer to an array of variables that have changed.
in	*NewVal*	Pointer to an array of new values for those variables.
in	*cVariables*	The count of variables included in this notification.

References genaNotifyAll(), GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UpnpPrintf(), and UpnpSdkInit.

Referenced by TvDeviceSetServiceTableVar().

7.1.5.46 UpnpNotifyExt()

```
int UpnpNotifyExt (
              UpnpDevice_Handle Hnd,
              const char * DevID,
              const char * ServID,
              IXML_Document * PropSet )
```

Similar to **UpnpNotify** except that it takes a DOM document for the event rather than an array of strings.

This function is synchronous and generates no callbacks.

This function may be called during a callback function to send out a notification.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid device handle.
- `UPNP_E_INVALID_SERVICE`: The **DevId/ServId pair** refers to an invalid service.
- `UPNP_E_INVALID_PARAM`: Either **VarName**, **NewVal**, **DevID**, **ServID**, or **PropSet** is not a valid pointer or **cVariables** is less than zero.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle to the device sending the event.
in	*DevID*	The device ID of the subdevice of the service generating the event.
in	*ServID*	The unique identifier of the service generating the event.
in	*PropSet*	The DOM document for the property set. Property set documents must conform to the XML schema defined in section 4.3 of the Universal Plug and Play Device Architecture specification.

References genaNotifyAllExt(), GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UpnpPrintf(), and UpnpSdkInit.

Referenced by TvDeviceSetServiceTableVar().

7.1.5.47 UpnpOpenHttpConnection()

```
int UpnpOpenHttpConnection (
            const char * url,
            void ** handle,
            int timeout )
```

Opens a connection to the server.

The SDK allocates the memory for **handle**, the application is responsible for freeing this memory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **url**, or **handle** is not a valid pointer.
- `UPNP_E_INVALID_URL`: The **url** is not a valid URL.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_SOCKET_ERROR`: Error occured allocating a socket and resources or an error occurred binding a socket.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*url*	The URL which contains the host, and the scheme to make the connection.
in,out	*handle*	A pointer in which to store the handle for this connection. This handle is required for futher operations over this connection.
in	*timeout*	The time out value sent with the request during which a response is expected from the receiver, failing which, an error is reported. If value is negative, timeout is infinite.

References http_OpenHttpConnection().

Referenced by UpnpOpenHttpGet(), and UpnpOpenHttpGetProxy().

7.1.5.48 UpnpOpenHttpGet()

```
int UpnpOpenHttpGet (
            const char * url,
            void ** handle,
            char ** contentType,
            int * contentLength,
            int * httpStatus,
            int timeout )
```

Gets a file specified in a URL.

The SDK allocates the memory for **handle** and **contentType**, the application is responsible for freeing this memory.

Note

Memory for **contentType** is freed when freeing the memory for handle.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: Either **url**, **handle**, is not a valid pointer.
- UPNP_E_INVALID_URL: The **url** is not a valid URL.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to download this file.
- UPNP_E_NETWORK_ERROR: A network error occurred.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket.
- UPNP_E_SOCKET_BIND: An error occurred binding a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting a socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.
- UPNP_E_BAD_RESPONSE: A bad response was received from the remote server.

Parameters

in	*url*	The URL of an item to get.
in,out	*handle*	A pointer to store the handle for this connection.
in,out	*contentType*	A buffer to store the media type of the item.
in,out	*contentLength*	A pointer to store the length of the item.
in,out	*httpStatus*	The status returned on receiving a response message.
in	*timeout*	The time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is

References UPNP_E_SUCCESS, UpnpEndHttpRequest(), UpnpGetHttpResponse(), UpnpMakeHttpRequest(), and UpnpOpenHttpConnection().

7.1.5.49 UpnpOpenHttpGetEx()

```
int UpnpOpenHttpGetEx (
            const char * url,
            void ** handle,
            char ** contentType,
            int * contentLength,
            int * httpStatus,
            int lowRange,
            int highRange,
            int timeout )
```

Gets specified number of bytes from a file specified in the URL.

The number of bytes is specified through a low count and a high count which are passed as a range of bytes for the request. The SDK allocates the memory for **handle** and **contentType**, the application is responsible for freeing this memory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **url**, **handle**, **contentType**, **contentLength** or **httpStatus** is not a valid pointer.
- `UPNP_E_INVALID_URL`: The **url** is not a valid URL.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_NETWORK_ERROR`: A network error occurred.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.
- `UPNP_E_BAD_RESPONSE`: A bad response was received from the remote server.

Parameters

`in`	*url*	The URL of the item to get.
`in,out`	*handle*	A pointer to store the handle for this connection.
`in,out`	*contentType*	A buffer to store the media type of the item.
`in,out`	*contentLength*	A buffer to store the length of the item.
`in,out`	*httpStatus*	The status returned on receiving a response message from the remote server.
`in`	*lowRange*	An integer value representing the low end of a range to retrieve.
`in`	*highRange*	An integer value representing the high end of a range to retrieve.
`in`	*timeout*	A time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is negative, timeout is infinite.

7.1.5.50 UpnpOpenHttpGetProxy()

```
int UpnpOpenHttpGetProxy (
             const char * url,
             const char * proxy_str,
             void ** handle,
             char ** contentType,
             int * contentLength,
             int * httpStatus,
             int timeout )
```

Gets a file specified in a URL through the specified proxy.

The SDK allocates the memory for **handle** and **contentType**, the application is responsible for freeing this memory.

Note

Memory for **contentType** is freed when freeing the memory for handle.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **url**, **handle**, is not a valid pointer.
- `UPNP_E_INVALID_URL`: The **url** is not a valid URL.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_NETWORK_ERROR`: A network error occurred.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.
- `UPNP_E_BAD_RESPONSE`: A bad response was received from the remote server.

Parameters

`in`	*url*	The URL of an item to get.
`in`	*proxy_str*	The URL of the proxy.
`in,out`	*handle*	A pointer to store the handle for this connection.
`in,out`	*contentType*	A buffer to store the media type of the item.
`in,out`	*contentLength*	A pointer to store the length of the item.
`in,out`	*httpStatus*	The status returned on receiving a response message.
`in`	*timeout*	The time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is negative, timeout is infinite.

References UPNP_E_SUCCESS, UpnpEndHttpRequest(), UpnpGetHttpResponse(), UpnpMakeHttpRequest(), and UpnpOpenHttpConnection().

7.1.5.51 UpnpOpenHttpPost()

```
int UpnpOpenHttpPost (
            const char * url,
            void ** handle,
            const char * contentType,
            int contentLength,
            int timeout )
```

Makes an HTTP POST request message, opens a connection to the server and sends the POST request to the server if the connection to the server succeeds.

The SDK allocates the memory for **handle**, the application is responsible for freeing this memory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **url**, **handle** or **contentType** is not a valid pointer.
- `UPNP_E_INVALID_URL`: The **url** is not a valid URL.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_SOCKET_ERROR`: Error occured allocating a socket and resources or an error occurred binding a socket.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

`in`	*url*	The URL in which to send the POST request.
`in,out`	*handle*	A pointer in which to store the handle for this connection. This handle is required for futher operations over this connection.
`in`	*contentType*	A buffer to store the media type of content being sent. Can be NULL.
`in`	*contentLength*	The length of the content, in bytes, being posted.
`in`	*timeout*	The time out value sent with the request during which a response is expected from the receiver, failing which, an error is reported. If value is negative, timeout is infinite.

References http_MakeHttpRequest(), http_OpenHttpConnection(), and UPNP_E_SUCCESS.

7.1.5.52 UpnpReadHttpGet()

```
int UpnpReadHttpGet (
            void * handle,
            char * buf,
            size_t * size,
            int timeout )
```

Gets specified number of bytes from a file specified in a URL.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, **buf** or **size** is not a valid pointer.
- `UPNP_E_BAD_RESPONSE`: A bad response was received from the remote server.
- `UPNP_E_BAD_HTTPMSG`: Either the request or response was in the incorrect format.
- `UPNP_E_CANCELED`: another thread called UpnpCancelHttpGet.

Note: In case of return values, the status code parameter of the passed in handle value may provide additional information on the return value.

Parameters

`in`	*handle*	The token created by the call to **UpnpOpenHttpGet**.
`in,out`	*buf*	The buffer to store the read item.
`in,out`	*size*	The size of the buffer to be read.
`in`	*timeout*	The time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is negative, timeout is infinite.

References http_ReadHttpResponse().

7.1.5.53 UpnpReadHttpResponse()

```
int UpnpReadHttpResponse (
            void * handle,
            char * buf,
            size_t * size,
            int timeout )
```

Reads the content of a response using a connection previously created by **UpnpOpenHttpConnection**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, **buf** or **size** is not a valid pointer.
- `UPNP_E_BAD_RESPONSE`: A bad response was received from the remote server.
- `UPNP_E_BAD_HTTPMSG`: Either the request or response was in the incorrect format.
- `UPNP_E_CANCELED`: another thread called UpnpCancelHttpGet.

Note: In case of return values, the status code parameter of the passed in handle value may provide additional information on the return value.

Parameters

`in`	*handle*	The handle of the connection created by the call to **UpnpOpenHttpConnection**.
`in,out`	*buf*	The buffer to store the read item.
`in,out`	*size*	The size of the buffer to be read.
`in`	*timeout*	The time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is negative, timeout is infinite.

References http_ReadHttpResponse().

7.1.5.54 UpnpRegisterClient()

```
int UpnpRegisterClient (
            Upnp_FunPtr Callback,
            const void * Cookie,
            UpnpClient_Handle * Hnd )
```

Registers a control point application with the UPnP Library.

A control point application cannot make any other API calls until it registers using this function.

UpnpRegisterClient is a synchronous call and generates no callbacks. Callbacks can occur as soon as this function returns.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_FINISH`: The SDK is already terminated or is not initialized.
- `UPNP_E_INVALID_PARAM`: Either **Callback** or **Hnd** is not a valid pointer.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to register this control point.

Parameters

in	*Callback*	Pointer to a function for receiving asynchronous events.
in	*Cookie*	Pointer to user data returned with the callback function when invoked.
out	*Hnd*	Pointer to a variable to store the new control point handle.

References Handle_Info::Callback, Handle_Info::ClientSubList, Handle_Info::Cookie, GetFreeHandle(), HandleTable, Handle_Info::HType, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, Handle_Info::SsdpSearchList, UPNP_E_ALREADY_REGISTERED, UPNP_E_FINISH, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_HANDLE, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkClientRegistered, UpnpSdkDeviceRegisteredV4, UpnpSdkDeviceregisteredV6, and UpnpSdkInit.

Referenced by TvCtrlPointStart().

7.1.5.55 UpnpRegisterRootDevice()

```
int UpnpRegisterRootDevice (
            const char * DescUrl,
            Upnp_FunPtr Callback,
            const void * Cookie,
            UpnpDevice_Handle * Hnd )
```

Registers a device application with the UPnP Library.

A device application cannot make any other API calls until it registers using this function.

Device applications can also register as control points (see **UpnpRegisterClient** to get a control point handle to perform control point functionality).

This is a synchronous call and does not generate any callbacks. Callbacks can occur as soon as this function returns.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_FINISH: The SDK is already terminated or is not initialized.
- UPNP_E_INVALID_DESC: The description document was not a valid device description.
- UPNP_E_INVALID_URL: The URL for the description document is not valid.
- UPNP_E_INVALID_PARAM: Either **Callback** or **Hnd** is not a valid pointer or **DescURL** is NULL.
- UPNP_E_NETWORK_ERROR: A network error occurred.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket.
- UPNP_E_SOCKET_BIND: An error occurred binding a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting the socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.
- UPNP_E_OUTOF_MEMORY: There are insufficient resources to register this root device.

Parameters

in	*DescUrl*	Pointer to a string containing the description URL for this root device instance.
in	*Callback*	Pointer to the callback function for receiving asynchronous events.
in	*Cookie*	Pointer to user data returned with the callback function when invoked.
out	*Hnd*	Pointer to a variable to store the new device handle.

References Handle_Info::aliasInstalled, Handle_Info::Callback, Handle_Info::ClientSubList, Handle_Info::Cookie, Handle_Info::DescDocument, Handle_Info::DescURL, Handle_Info::DeviceAf, Handle_Info::DeviceList, FreeHandle(), GetFreeHandle(), HandleTable, Handle_Info::HType, ixmlDocument_free(), ixmlDocument_getElementsByTagName(), Handle_Info::LowerDescURL, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, Handle_Info::ServiceList, Handle_Info::ServiceTable, Handle_Info::SsdpSearchList, UPNP_E_FINISH, UPNP_E_INVALID_DESC, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_HANDLE, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpDownloadXmlDoc(), UpnpPrintf(), UpnpSdkDeviceRegisteredV4, and UpnpSdkInit.

7.1.5.56 UpnpRegisterRootDevice2()

```
int UpnpRegisterRootDevice2 (
            Upnp_DescType descriptionType,
            const char * description,
            size_t bufferLen,
            int config_baseURL,
            Upnp_FunPtr Fun,
            const void * Cookie,
            UpnpDevice_Handle * Hnd )
```

Registers a device application with the UPnP Library. Similar to **UpnpRegisterRootDevice**, except that it also allows the description document to be specified as a file or a memory buffer.

The description can also be configured to have the correct IP and port address.

NOTE: For the configuration to be functional, the internal web server MUST be present. In addition, the web server MUST be activated (using **UpnpSetWebServerRootDir**) before calling this function. The only condition where the web server can be absent is if the description document is specified as a URL and no configuration is required (i.e. config_baseURL = 0.)

This is a synchronous call and does not generate any callbacks. Callbacks can occur as soon as this function returns.

Examples of using different types of description documents:

```
1) Description specified as a URL:
      descriptionType == UPNPREG_URL_DESC
      description is the URL
      bufferLen = 0 (ignored)
2) Description specified as a file:
      descriptionType == UPNPREG_FILENAME_DESC
      description is a filename
      bufferLen = 0 (ignored)
3) Description specified as a memory buffer:
      descriptionType == UPNPREG_BUF_DESC
      description is pointer to a memory buffer
      bufferLen == length of memory buffer
```

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_FINISH`: The SDK is already terminated or is not initialized.
- `UPNP_E_INVALID_DESC`: The description document is not a valid device description.
- `UPNP_E_INVALID_PARAM`: Either **Callback** or **Hnd** is not a valid pointer or **DescURL** is `NULL`.
- `UPNP_E_NETWORK_ERROR`: A network error occurred.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting the socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.
- `UPNP_E_OUTOF_MEMORY`: There are insufficient resources to register this root device.
- `UPNP_E_URL_TOO_BIG`: Length of the URL is bigger than the internal buffer.
- `UPNP_E_FILE_NOT_FOUND`: The description file could not be found.
- `UPNP_E_FILE_READ_ERROR`: An error occurred reading the description file.
- `UPNP_E_INVALID_URL`: The URL to the description document is invalid.
- `UPNP_E_EXT_NOT_XML`: The URL to the description document or file should have a `.xml` extension.
- `UPNP_E_NO_WEB_SERVER`: The internal web server has been compiled out; the SDK cannot configure itself from the description document.

Parameters

`in`	*descriptionType*	The type of the description document.
`in`	*description*	Treated as a URL, file name or memory buffer depending on description type.
`in`	*bufferLen*	The length of memory buffer if passing a description in a buffer, otherwise it is ignored.
`in`	*config_baseURL*	If nonzero, `URLBase` of description document is configured and the description is served using the internal web server.
`in`	*Fun*	Pointer to the callback function for receiving asynchronous events.
`in`	*Cookie*	Pointer to user data returned with the callback function when invoked.
`out`	*Hnd*	Pointer to a variable to store the new device handle.

References Handle_Info::aliasInstalled, Handle_Info::Callback, Handle_Info::ClientSubList, Handle_Info::Cookie, Handle_Info::DescDocument, Handle_Info::DescURL, Handle_Info::DeviceAf, Handle_Info::DeviceList, FreeHandle(), GetDescDocumentAndURL(), GetFreeHandle(), HandleTable, Handle_Info::HType, ixmlDocument_free(), ixmlDocument_getElementsByTagName(), Handle_Info::LowerDescURL, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, Handle_Info::ServiceList, Handle_Info::ServiceTable, Handle_Info::SsdpSearchList, UPNP_E_FINISH, UPNP_E_INVALID_DESC, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_HANDLE, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkDeviceRegisteredV4, and UpnpSdkInit.

7.1.5.57 UpnpRegisterRootDevice3()

```
int UpnpRegisterRootDevice3 (
            const char * DescUrl,
            Upnp_FunPtr Callback,
            const void * Cookie,
            UpnpDevice_Handle * Hnd,
            int AddressFamily )
```

Registers a device application for a specific address family with the UPnP library.

A device application cannot make any other API calls until it registers using this function. Device applications can also register as control points (see **UpnpRegisterClient** to get a control point handle to perform control point functionality).

This is synchronous and does not generate any callbacks. Callbacks can occur as soon as this function returns.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_FINISH`: The SDK is already terminated or is not initialized.
- `UPNP_E_INVALID_DESC`: The description document was not a valid device description.
- `UPNP_E_INVALID_URL`: The URL for the description document is not valid.
- `UPNP_E_INVALID_PARAM`: Either **Callback** or **Hnd** is not a valid pointer or **DescURL** is `NULL`.
- `UPNP_E_NETWORK_ERROR`: A network error occurred.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting the socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.
- `UPNP_E_OUTOF_MEMORY`: There are insufficient resources to register this root device.

Parameters

`in`	*DescUrl*	Pointer to a string containing the description URL for this root device instance.
`in`	*Callback*	Pointer to the callback function for receiving asynchronous events.
`in`	*Cookie*	Pointer to user data returned with the callback function when invoked.
`out`	*Hnd*	Pointer to a variable to store the new device handle.
`in`	*AddressFamily*	Address family of this device. Can be AF_INET for an IPv4 device, or AF_INET6 for an IPv6 device. Defaults to AF_INET.

References Handle_Info::Cookie, UpnpPrintf(), and UpnpRegisterRootDevice4().

Referenced by TvDeviceStart().

7.1.5.58 UpnpRegisterRootDevice4()

```
int UpnpRegisterRootDevice4 (
            const char * DescUrl,
            Upnp_FunPtr Callback,
            const void * Cookie,
            UpnpDevice_Handle * Hnd,
            int AddressFamily,
            const char * LowerDescUrl )
```

Registers a device application for a specific address family with the UPnP library. This function can also be used to specify a dedicated description URL to be returned for legacy CPs.

A device application cannot make any other API calls until it registers using this function. Device applications can also register as control points (see **UpnpRegisterClient** to get a control point handle to perform control point functionality).

This is synchronous and does not generate any callbacks. Callbacks can occur as soon as this function returns.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_FINISH`: The SDK is already terminated or is not initialized.
- `UPNP_E_INVALID_DESC`: The description document was not a valid device description.
- `UPNP_E_INVALID_URL`: The URL for the description document is not valid.
- `UPNP_E_INVALID_PARAM`: Either **Callback** or **Hnd** is not a valid pointer or **DescURL** is `NULL`.
- `UPNP_E_NETWORK_ERROR`: A network error occurred.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting the socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.
- `UPNP_E_OUTOF_MEMORY`: There are insufficient resources to register this root device.

Parameters

in	*DescUrl*	Pointer to a string containing the description URL for this root device instance.
in	*Callback*	Pointer to the callback function for receiving asynchronous events.
in	*Cookie*	Pointer to user data returned with the callback function when invoked.
out	*Hnd*	Pointer to a variable to store the new device handle.
in	*AddressFamily*	Address family of this device. Can be AF_INET for an IPv4 device, or AF_INET6 for an IPv6 device. Defaults to AF_INET.
in	*LowerDescUrl*	Pointer to a string containing the description URL to be returned for legacy CPs for this root device instance.

References Handle_Info::aliasInstalled, Handle_Info::Callback, Handle_Info::ClientSubList, Handle_Info::Cookie, Handle_Info::DescDocument, Handle_Info::DescURL, Handle_Info::DeviceAf, Handle_Info::DeviceList, FreeHandle(), GetFreeHandle(), HandleTable, Handle_Info::HType, ixmlDocument_free(), ixmlDocument_getElementsByTagName(), Handle_Info::LowerDescURL, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, Handle_Info::ServiceList, Handle_Info::ServiceTable, Handle_Info::SsdpSearchList, UPNP_E_FINISH, UPNP_E_INVALID_DESC, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_HANDLE, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpDownloadXmlDoc(), UpnpPrintf(), UpnpSdkDeviceRegisteredV4, UpnpSdkDeviceregisteredV6, and UpnpSdkInit.

Referenced by UpnpRegisterRootDevice3().

7.1.5.59 UpnpRemoveAllVirtualDirs()

```
void UpnpRemoveAllVirtualDirs (
            void  )
```

Removes all virtual directory mappings.

References pVirtualDirList, and UpnpSdkInit.

Referenced by UpnpFinish().

7.1.5.60 UpnpRemoveVirtualDir()

```
int UpnpRemoveVirtualDir (
            const char * dirName )
```

Removes a virtual directory mapping made with **UpnpAddVirtualDir**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **dirName** is not valid.

Parameters

in	*dirName*	The name of the virtual directory mapping to remove.

References pVirtualDirList, UPNP_E_FINISH, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpSdkInit.

7.1.5.61 UpnpRenewSubscription()

```
int UpnpRenewSubscription (
            UpnpClient_Handle Hnd,
            int * TimeOut,
            const Upnp_SID SubsId )
```

Renews a subscription that is about to expire.

This function is synchronous.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_PARAM: **Timeout** is not a valid pointer.
- UPNP_E_INVALID_SID: The SID being passed to this function is not a valid subscription ID.
- UPNP_E_NETWORK_ERROR: A network error occured.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket.
- UPNP_E_SOCKET_BIND: An error occurred binding a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting to **PublisherUrl**.
- UPNP_E_OUTOF_SOCKET: An error occurred creating a socket.
- UPNP_E_BAD_RESPONSE: An error occurred in response from the publisher.
- UPNP_E_SUBSCRIBE_UNACCEPTED: The publisher refused the subscription renew.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the control point that is renewing the subscription.
in,out	*TimeOut*	Pointer to a variable containing the requested subscription time. Upon return, it contains the actual renewal time.
in	*SubsId*	The ID for the subscription to renew.

References genaRenewSubscription(), GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UpnpPrintf(), UpnpSdkInit, UpnpString_delete(), UpnpString_new(), and UpnpString_set_String().

7.1.5.62 UpnpRenewSubscriptionAsync()

```
int UpnpRenewSubscriptionAsync (
            UpnpClient_Handle Hnd,
            int TimeOut,
            Upnp_SID SubsId,
            Upnp_FunPtr Fun,
            const void * Cookie )
```

Renews a subscription that is about to expire, generating a callback when the operation is complete.

Note that many of the error codes for this function are returned in the **UpnpEventSubscribe** structure. In those cases, the function returns UPNP_E_SUCCESS and the appropriate error code will be in the **UpnpEvent↩ Subscribe.ErrCode** field in the **Event** structure passed to the callback.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_SID: The **SubsId** is not a valid subscription ID.

- UPNP_E_INVALID_PARAM: Either **Fun** is not a valid callback function pointer or **Timeout** is less than zero but is not UPNP_INFINITE.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.
- UPNP_E_NETWORK_ERROR: A network error occured (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_BIND: An error occurred binding the socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_CONNECT: An error occurred connecting to **PublisherUrl** (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_OUTOF_SOCKET: An error occurred creating socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_BAD_RESPONSE: An error occurred in response from the publisher (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SUBSCRIBE_UNACCEPTED: The publisher refused the subscription request (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).

Parameters

in	*Hnd*	The handle of the control point that is renewing the subscription.
in	*TimeOut*	The requested subscription time. The actual timeout value is returned when the callback function is called.
in	*SubsId*	The ID for the subscription to renew.
in	*Fun*	Pointer to a callback function to be invoked when the renewal is complete.
in	*Cookie*	Pointer to user data passed to the callback function when invoked.

References GetHandleInfo(), gSendThreadPool, ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkInit, and UpnpThreadDistribution().

7.1.5.63 UpnpSearchAsync()

```
int UpnpSearchAsync (
            UpnpClient_Handle Hnd,
            int Mx,
            const char * TTarget_constarget_const,
            const void * Cookie_const )
```

Searches for devices matching the given search target.

The function returns immediately and the SDK calls the default callback function, registered during the **UpnpRegisterClient** call, for each matching root device, device, or service. The application specifies the search type by the **Target** parameter.

This function searches for the devices for the provided maximum time. It is an asynchronous function. It schedules a search job and returns. The client is notified about the search results after search timer.

Note that there is no way for the SDK to distinguish which client instance issued a particular search. Therefore, the client can get search callbacks that do not match the original criteria of the search. Also, the application will receive multiple callbacks for each search.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_PARAM: **Target** is NULL.

Parameters

Hnd	The handle of the client performing the search.
Mx	The time, in seconds, to wait for responses. If the time is greater than MAX_SEARCH_TIME then the time is set to MAX_SEARCH_TIME. If the time is less than MIN_SEARCH_TIME then the time is set to MIN_SEARCH_TIME.
TTarget_constarget_const	The search target as defined in the UPnP Device Architecture v1.0 specification.
Cookie_const	The user data to pass when the callback function is invoked.

References GetHandleInfo(), SearchByTarget(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UpnpPrintf(), and UpnpSdkInit.

Referenced by TvCtrlPointVerifyTimeouts().

7.1.5.64 UpnpSendAction()

```
int UpnpSendAction (
            UpnpClient_Handle Hnd,
            const char * ActionURL,
            const char * ServiceType,
            const char * DevUDN,
            IXML_Document * Action,
            IXML_Document ** RespNode )
```

Sends a message to change a state variable in a service.

This is a synchronous call that does not return until the action is complete.

Note that a positive return value indicates a SOAP-protocol error code. In this case, the error description can be retrieved from **RespNode**. A negative return value indicates an SDK error.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_URL: **ActionUrl** is not a valid URL.
- UPNP_E_INVALID_ACTION: This action is not valid.
- UPNP_E_INVALID_DEVICE: **DevUDN** is not a valid device.
- UPNP_E_INVALID_PARAM: **ServiceType**, **Action**, **ActionUrl**, or **RespNode** is not a valid pointer.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the control point sending the action.
in	*ActionURL*	The action URL of the service.
in	*ServiceType*	The type of the service.
in	*DevUDN*	This parameter is ignored and must be NULL.
in	*Action*	The DOM document for the action.
out	*RespNode*	The DOM document for the response to the action. The SDK allocates this document and the caller needs to free it.

References GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UpnpPrintf(), and UpnpSdkInit.

Referenced by UpnpSendActionEx().

7.1.5.65 UpnpSendActionAsync()

```
int UpnpSendActionAsync (
            UpnpClient_Handle Hnd,
            const char * ActionURL,
            const char * ServiceType,
            const char * DevUDN,
            IXML_Document * Action,
            Upnp_FunPtr Fun,
            const void * Cookie )
```

Sends a message to change a state variable in a service, generating a callback when the operation is complete.

See **UpnpSendAction** for comments on positive return values. These positive return values are sent in the event struct associated with the UPNP_CONTROL_ACTION_COMPLETE event.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_URL: **ActionUrl** is an invalid URL.
- UPNP_E_INVALID_DEVICE: **DevUDN** is an invalid device.
- UPNP_E_INVALID_PARAM: Either **Fun** is not a valid callback function or **ServiceType**, **Act**, or **ActionUrl** is NULL.
- UPNP_E_INVALID_ACTION: This action is not valid.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the control point sending the action.
in	*ActionURL*	The action URL of the service.
in	*ServiceType*	The type of the service.
in	*DevUDN*	This parameter is ignored and must be NULL.
in	*Action*	The DOM document for the action to perform on this device.
in	*Fun*	Pointer to a callback function to be invoked when the operation completes.
in	*Cookie*	Pointer to user data that to be passed to the callback when invoked.

References DOMString, free_action_arg(), GetHandleInfo(), gSendThreadPool, ixmlFreeDOMString(), ixmlParseBufferEx(), ixmlPrintNode(), ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_FINISH, UPNP_E_INVALID_ACTION, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkInit, and UpnpThreadDistribution().

Referenced by UpnpSendActionExAsync().

7.1.5.66 UpnpSendActionEx()

```
int UpnpSendActionEx (
            UpnpClient_Handle Hnd,
            const char * ActionURL,
            const char * ServiceType,
            const char * DevUDN,
            IXML_Document * Header,
            IXML_Document * Action,
            IXML_Document ** RespNode )
```

Sends a message to change a state variable in a service.

This is a synchronous call that does not return until the action is complete.

Note that a positive return value indicates a SOAP-protocol error code. In this case, the error description can be retrieved from **RespNode**. A negative return value indicates an SDK error.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid control point handle.
- `UPNP_E_INVALID_URL`: **ActionUrl** is not a valid URL.
- `UPNP_E_INVALID_ACTION`: This action is not valid.
- `UPNP_E_INVALID_DEVICE`: **DevUDN** is not a valid device.
- `UPNP_E_INVALID_PARAM`: **ServiceType**, **Action**, **ActionUrl**, or **RespNode** is not a valid pointer.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the control point sending the action.
in	*ActionURL*	The action URL of the service.
in	*ServiceType*	The type of the service.
in	*DevUDN*	This parameter is ignored and must be `NULL`.
in	*Header*	The DOM document for the SOAP header. This may be `NULL` if the header is not required.
in	*Action*	The DOM document for the action.
out	*RespNode*	The DOM document for the response to the action. The SDK allocates this document and the caller needs to free it.

References GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UpnpPrintf(), UpnpSdkInit, and UpnpSendAction().

7.1.5.67 UpnpSendActionExAsync()

```
int UpnpSendActionExAsync (
            UpnpClient_Handle Hnd,
            const char * ActionURL,
            const char * ServiceType,
            const char * DevUDN,
            IXML_Document * Header,
            IXML_Document * Action,
            Upnp_FunPtr Fun,
            const void * Cookie )
```

Sends a message to change a state variable in a service, generating a callback when the operation is complete.

See **UpnpSendAction** for comments on positive return values. These positive return values are sent in the event struct associated with the UPNP_CONTROL_ACTION_COMPLETE event.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_URL: **ActionUrl** is an invalid URL.
- UPNP_E_INVALID_DEVICE: **DevUDN** is an invalid device.
- UPNP_E_INVALID_PARAM: Either **Fun** is not a valid callback function or **ServiceType**, **Act**, or **ActionUrl** is NULL.
- UPNP_E_INVALID_ACTION: This action is not valid.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the control point sending the action.
in	*ActionURL*	The action URL of the service.
in	*ServiceType*	The type of the service.
in	*DevUDN*	This parameter is ignored and must be NULL.
in	*Header*	The DOM document for the SOAP header. This may be NULL if the header is not required.
in	*Action*	The DOM document for the action to perform on this device.
in	*Fun*	Pointer to a callback function to be invoked when the operation completes.
in	*Cookie*	Pointer to user data that to be passed to the callback when invoked.

References DOMString, free_action_arg(), GetHandleInfo(), gSendThreadPool, ixmlDocument_free(), ixmlFreeDOMString(), ixmlParseBufferEx(), ixmlPrintNode(), ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_FINISH, UPNP_E_INVALID_ACTION, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkInit, UpnpSendActionAsync(), and UpnpThreadDistribution().

7.1.5.68 UpnpSendAdvertisement()

```
int UpnpSendAdvertisement (
            UpnpDevice_Handle Hnd,
            int Exp )
```

Sends out the discovery announcements for all devices and services for a device.

Each announcement is made with the same expiration time.

This is a synchronous call.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid device handle.
- UPNP_E_OUTOF_MEMORY: There are insufficient resources to send future advertisements.

Parameters

Hnd	The device handle for which to send out the announcements.
Exp	The expiration age, in seconds, of the announcements. If the expiration age is less than 1 then the expiration age is set to DEFAULT_MAXAGE. If the expiration age is less than or equal to AUTO_ADVERTISEMENT_TIME * 2 then the expiration age is set to (AUTO_ADVERTISEMENT_TIME + 1) * 2.

References UpnpPrintf(), and UpnpSendAdvertisementLowPower().

Referenced by AutoAdvertise(), and TvDeviceStart().

7.1.5.69 UpnpSendAdvertisementLowPower()

```
int UpnpSendAdvertisementLowPower (
            UpnpDevice_Handle Hnd,
            int Exp,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Sends out the discovery announcements for all devices and services for a device.

Each announcement is made with the same expiration time.

This is a synchronous call.

This function allow a device to specify the SSDP extensions defined by UPnP Low Power.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid device handle.
- UPNP_E_OUTOF_MEMORY: There are insufficient resources to send future advertisements.

Parameters

Hnd	The device handle for which to send out the announcements.
Exp	The expiration age, in seconds, of the announcements. If the expiration age is less than 1 then the expiration age is set to DEFAULT_MAXAGE. If the expiration age is less than or equal to AUTO_ADVERTISEMENT_TIME * 2 then the expiration age is set to (AUTO_ADVERTISEMENT_TIME + 1) * 2.
PowerState	PowerState as defined by UPnP Low Power.
SleepPeriod	SleepPeriod as defined by UPnP Low Power.
RegistrationState	RegistrationState as defined by UPnP Low Power.

References AdvertiseAndReply(), AutoAdvertise(), free_advertise_arg(), GetHandleInfo(), gTimerThread, REL_SEC, TimerThreadSchedule(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), and UpnpSdkInit.

Referenced by UpnpSendAdvertisement().

7.1.5.70 UpnpSetContentLength()

```
int UpnpSetContentLength (
            UpnpClient_Handle Hnd,
            size_t contentLength )
```

Deprecated Use **UpnpSetMaxContentLength** instead.

Warning: the Handle argument provided here is not used, so the effect of this function is global to the SDK (= same as **UpnpSetMaxContentLength**).

Parameters

in	*Hnd*	The handle of the device instance for which the coincoming content length needs to be set.
in	*contentLength*	Permissible content length

References g_maxContentLength, GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, and UpnpSdkInit.

7.1.5.71 UpnpSetMaxContentLength()

```
int UpnpSetMaxContentLength (
            size_t contentLength )
```

Sets the maximum content-length that the SDK will process on an incoming SOAP requests or responses.

This API allows devices that have memory constraints to exhibit consistent behaviour if the size of the incoming SOAP message exceeds the memory that device can allocate.

If set to 0 then checking will be disabled.

The default maximum content-length is DEFAULT_SOAP_CONTENT_LENGTH = 16K bytes.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.

Parameters

in	*contentLength*	The maximum permissible content length for incoming SOAP actions, in bytes.

References g_maxContentLength, UPNP_E_FINISH, UPNP_E_SUCCESS, and UpnpSdkInit.

7.1.5.72 UpnpSetMaxSubscriptions()

```
int UpnpSetMaxSubscriptions (
            UpnpDevice_Handle Hnd,
            int MaxSubscriptions )
```

Sets the maximum number of subscriptions accepted per service.

The default value accepts as many as system resources allow. If the number of current subscriptions for a service is greater than the requested value, the SDK accepts no new subscriptions or renewals, however, the SDK does not remove any current subscriptions.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid device handle.

Parameters

Hnd	The handle of the device for which the maximum number of subscriptions is being set.
MaxSubscriptions	The maximum number of subscriptions to be allowed per service.

References GetHandleInfo(), Handle_Info::MaxSubscriptions, UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, UpnpPrintf(), and UpnpSdkInit.

7.1.5.73 UpnpSetMaxSubscriptionTimeOut()

```
int UpnpSetMaxSubscriptionTimeOut (
            UpnpDevice_Handle Hnd,
            int MaxSubscriptionTimeOut )
```

Sets the maximum time-out accepted for a subscription request or renewal.

The default value accepts the time-out set by the control point. If a control point requests a subscription time-out less than or equal to the maximum, the SDK grants the value requested by the control point. If the time-out is greater, the SDK returns the maximum value.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid device handle.

Parameters

Hnd	The handle of the device for which the maximum subscription time-out is being set.
MaxSubscriptionTimeOut	The maximum subscription time-out to be accepted.

References GetHandleInfo(), Handle_Info::MaxSubscriptionTimeOut, UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, UpnpPrintf(), and UpnpSdkInit.

7.1.5.74 UpnpSetWebServerCorsString()

```
int UpnpSetWebServerCorsString (
            const char * corsString )
```

Assign the Access-Control-Allow-Origin specfied by the input const char* cors_string parameterto the global CORS string.

Note

This function is not available when the web server is not compiled into the UPnP Library.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **rootDir** is an invalid directory.

Parameters

in	*corsString*	String having the Access-Control-Allow-Origin string.

7.1.5.75 UpnpSetWebServerRootDir()

```
int UpnpSetWebServerRootDir (
            const char * rootDir )
```

Sets the document root directory for the internal web server.

This directory is considered the root directory (i.e. "/") of the web server.

This function also activates or deactivates the web server. To disable the web server, pass `NULL` for **rootDir**; to activate, pass a valid directory string.

Note

This function is not available when the web server is not compiled into the UPnP Library.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **rootDir** is an invalid directory.

Parameters

in	*rootDir*	Path of the root directory of the web server.

Referenced by TvDeviceStart().

7.1.5.76 UpnpSubscribe()

```
int UpnpSubscribe (
            UpnpClient_Handle Hnd,
            const char * PublisherUrl,
            int * TimeOut,
            Upnp_SID SubsId )
```

Registers a control point to receive event notifications from another device.

This operation is synchronous.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid control point handle.
- `UPNP_E_INVALID_URL`: **PublisherUrl** is not a valid URL.
- `UPNP_E_INVALID_PARAM`: **Timeout** is not a valid pointer or **SubsId** or **PublisherUrl** is `NULL`.
- `UPNP_E_NETWORK_ERROR`: A network error occured.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting to **PublisherUrl**.
- `UPNP_E_OUTOF_SOCKET`: An error occurred creating a socket.
- `UPNP_E_BAD_RESPONSE`: An error occurred in response from the publisher.
- `UPNP_E_SUBSCRIBE_UNACCEPTED`: The publisher refused the subscription request.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the control point.
in	*PublisherUrl*	The URL of the service to subscribe to.
in,out	*TimeOut*	Pointer to a variable containing the requested subscription time. Upon return, it contains the actual subscription time returned from the service.
out	*SubsId*	Pointer to a variable to receive the subscription ID (SID).

References genaSubscribe(), GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UpnpPrintf(), UpnpSdkInit, UpnpString_delete(), UpnpString_get_String(), UpnpString_new(), and UpnpString_set_String().

7.1.5.77 UpnpSubscribeAsync()

```
int UpnpSubscribeAsync (
            UpnpClient_Handle Hnd,
            const char * PublisherUrl,
            int TimeOut,
            Upnp_FunPtr Fun,
            const void * Cookie )
```

Performs the same operation as **UpnpSubscribe**, but returns immediately and calls the registered callback function when the operation is complete.

Note that many of the error codes for this function are returned in the **UpnpEventSubscribe** structure. In those cases, the function returns UPNP_E_SUCCESS and the appropriate error code will be in the **UpnpEvent↩ Subscribe.ErrCode** field in the **Event** structure passed to the callback.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_URL: The **PublisherUrl** is not a valid URL.
- UPNP_E_INVALID_PARAM: Either **TimeOut** or **Fun** or **PublisherUrl** is not a valid pointer.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.
- UPNP_E_NETWORK_ERROR: A network error occured (returned in the **UpnpEventSubscribe.Err↩ Code** field as part of the callback).
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket (returned in the **Upnp↩ EventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket (returned in the **Upnp↩ EventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_BIND: An error occurred binding the socket (returned in the **UpnpEvent↩ Subscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_CONNECT: An error occurred connecting to **PublisherUrl** (returned in the **Upnp↩ EventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_OUTOF_SOCKET: An error occurred creating the socket (returned in the **UpnpEvent↩ Subscribe.ErrCode** field as part of the callback).
- UPNP_E_BAD_RESPONSE: An error occurred in response from the publisher (returned in the **Upnp↩ EventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SUBSCRIBE_UNACCEPTED: The publisher refused the subscription request (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).

Parameters

Hnd	The handle of the control point that is subscribing.
PublisherUrl	The URL of the service to subscribe to.
TimeOut	The requested subscription time. Upon return, it contains the actual subscription time returned from the service.
Fun	Pointer to the callback function for this subscribe request.
Cookie	A user data value passed to the callback function when invoked.

References GetHandleInfo(), gSendThreadPool, ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkInit, and UpnpThreadDistribution().

7.1.5.78 UpnpThreadDistribution()

```
void UpnpThreadDistribution (
            struct UpnpNonblockParam * Param )
```

Schedule async functions in threadpool.

References DOMString, free_action_arg(), genaRenewSubscription(), genaSubscribe(), genaUnSubscribe(), ixmlDocument_free(), UPNP_CONTROL_ACTION_COMPLETE, UPNP_CONTROL_GET_VAR_COMPLETE, UPNP_EVENT_RENEWAL_COMPLETE, UPNP_EVENT_SUBSCRIBE_COMPLETE, UPNP_EVENT_UNSUBSCRIBE_COMPLETE, UpnpActionComplete_delete(), UpnpActionComplete_new(), UpnpActionComplete_set_ActionRequest(), UpnpActionComplete_set_, UpnpActionComplete_set_ErrCode(), UpnpActionComplete_strcpy_CtrlUrl(), UpnpEventSubscribe_delete(), UpnpEventSubscribe_get_PublisherUrl(), UpnpEventSubscribe_get_SID(), UpnpEventSubscribe_new(), UpnpEventSubscribe_set_E, UpnpEventSubscribe_set_SID(), UpnpEventSubscribe_set_TimeOut(), UpnpEventSubscribe_strcpy_PublisherUrl(), UpnpEventSubscribe_strcpy_SID(), UpnpPrintf(), UpnpStateVarComplete_delete(), UpnpStateVarComplete_new(), UpnpStateVarComplete_set_CurrentVal(), UpnpStateVarComplete_set_ErrCode(), UpnpStateVarComplete_strcpy_CtrlUrl(), UpnpStateVarComplete_strcpy_StateVarName(), UpnpString_delete(), and UpnpString_new().

Referenced by UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

7.1.5.79 UpnpUnRegisterClient()

```
int UpnpUnRegisterClient (
            UpnpClient_Handle Hnd )
```

Unregisters a control point application, unsubscribing all active subscriptions.

This function unregisters a client registered with UpnpRegisterClient. After this call, the **UpnpClient_Handle** is no longer valid. The UPnP Library generates no more callbacks after this function returns.

UpnpUnRegisterClient is a synchronous call and generates no callbacks.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid control point handle.

Parameters

in	*Hnd*	The handle of the control point instance to unregister.

References FreeHandle(), genaUnregisterClient(), GetHandleInfo(), Handle_Info::SsdpSearchList, UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkClientRegistered, and UpnpSdkInit.

Referenced by UpnpFinish().

7.1.5.80 UpnpUnRegisterRootDevice()

```
int UpnpUnRegisterRootDevice (
            UpnpDevice_Handle Hnd )
```

Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **Upnp↩ RegisterRootDevice3** or **UpnpRegisterRootDevice4**.

After this call, the **UpnpDevice_Handle** is no longer valid. For all advertisements that have not yet expired, the SDK sends a device unavailable message automatically.

This is a synchronous call and generates no callbacks. Once this call returns, the SDK will no longer generate callbacks to the application.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid device handle.

Parameters

in	*Hnd*	The handle of the root device instance to unregister.

References UpnpPrintf(), and UpnpUnRegisterRootDeviceLowPower().

Referenced by TvDeviceStop(), and UpnpFinish().

7.1.5.81 UpnpUnRegisterRootDeviceLowPower()

```
int UpnpUnRegisterRootDeviceLowPower (
            UpnpDevice_Handle Hnd,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **Upnp↩ RegisterRootDevice3** or **UpnpRegisterRootDevice4**.

After this call, the **UpnpDevice_Handle** is no longer valid. For all advertisements that have not yet expired, the SDK sends a device unavailable message automatically.

This is a synchronous call and generates no callbacks. Once this call returns, the SDK will no longer generate callbacks to the application.

This function allow a device to specify the SSDP extensions defined by UPnP Low Power.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid device handle.

Parameters

in	*Hnd*	The handle of the root device instance to unregister.
	PowerState	PowerState as defined by UPnP Low Power.
	SleepPeriod	SleepPeriod as defined by UPnP Low Power.
	RegistrationState	RegistrationState as defined by UPnP Low Power.

References AdvertiseAndReply(), Handle_Info::aliasInstalled, Handle_Info::DescDocument, Handle_Info::DeviceAf, Handle_Info::DeviceList, FreeHandle(), genaUnregisterDevice(), GetHandleInfo(), ixmlDocument_free(), ixmlNodeList_free(), Handle_Info::ServiceList, Handle_Info::SsdpSearchList, UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkDeviceRegisteredV4, UpnpSdkDeviceregisteredV6, UpnpSdkInit, and web_server_set_alias().

Referenced by UpnpUnRegisterRootDevice().

7.1.5.82 UpnpUnSubscribe()

```
int UpnpUnSubscribe (
            UpnpClient_Handle Hnd,
            const Upnp_SID SubsId )
```

Removes the subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**.

This is a synchronous call.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_HANDLE`: The handle is not a valid control point handle.
- `UPNP_E_INVALID_SID`: The **SubsId** is not a valid subscription ID.
- `UPNP_E_NETWORK_ERROR`: A network error occured.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_SOCKET_BIND`: An error occurred binding a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting to **PublisherUrl**.
- `UPNP_E_OUTOF_SOCKET`: An error ocurred creating a socket.
- `UPNP_E_BAD_RESPONSE`: An error occurred in response from the publisher.
- `UPNP_E_UNSUBSCRIBE_UNACCEPTED`: The publisher refused the unsubscribe request (the client is still unsubscribed and no longer receives events).
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

in	*Hnd*	The handle of the subscribed control point.
in	*SubsId*	The ID returned when the control point subscribed to the service.

References genaUnSubscribe(), GetHandleInfo(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UpnpPrintf(), UpnpSdkInit, UpnpString_delete(), UpnpString_new(), and UpnpString_set_String().

7.1.5.83 UpnpUnSubscribeAsync()

```
int UpnpUnSubscribeAsync (
              UpnpClient_Handle Hnd,
              Upnp_SID SubsId,
              Upnp_FunPtr Fun,
              const void * Cookie )
```

Removes a subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**, generating a callback when the operation is complete.

Note that many of the error codes for this function are returned in the **UpnpEventSubscribe** structure. In those cases, the function returns UPNP_E_SUCCESS and the appropriate error code will be in the **UpnpEventSubscribe.ErrCode** field in the **Event** structure passed to the callback.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_HANDLE: The handle is not a valid control point handle.
- UPNP_E_INVALID_SID: The **SubsId** is not a valid SID.
- UPNP_E_INVALID_PARAM: **Fun** is not a valid callback function pointer.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.
- UPNP_E_NETWORK_ERROR: A network error occured (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_BIND: An error occurred binding the socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_SOCKET_CONNECT: An error occurred connecting to **PublisherUrl** (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_OUTOF_SOCKET: An error occurred creating a socket (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_BAD_RESPONSE: An error occurred in response from the publisher (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).
- UPNP_E_UNSUBSCRIBE_UNACCEPTED: The publisher refused the subscription request (returned in the **UpnpEventSubscribe.ErrCode** field as part of the callback).

Parameters

in	*Hnd*	The handle of the subscribed control point.
in	*SubsId*	The ID returned when the control point subscribed to the service.
in	*Fun*	Pointer to a callback function to be called when the operation is complete.
in	*Cookie*	Pointer to user data to pass to the callback function when invoked.

References GetHandleInfo(), gSendThreadPool, ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_FINISH, UPNP_E_INVALID_HANDLE, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpSdkInit, and UpnpThreadDistribution().

7.1.5.84 UpnpVirtualDir_set_CloseCallback()

```
int UpnpVirtualDir_set_CloseCallback (
            VDCallback_Close callback )
```

Sets the close callback function to be used to access a virtual directory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **callback** is not a valid pointer.

References VirtualDirCallbacks::close, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and virtualDirCallback.

7.1.5.85 UpnpVirtualDir_set_GetInfoCallback()

```
int UpnpVirtualDir_set_GetInfoCallback (
            VDCallback_GetInfo callback )
```

Sets the get_info callback function to be used to access a virtual directory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **callback** is not a valid pointer.

References VirtualDirCallbacks::get_info, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and virtualDirCallback.

7.1.5.86 UpnpVirtualDir_set_OpenCallback()

```
int UpnpVirtualDir_set_OpenCallback (
            VDCallback_Open callback )
```

Sets the open callback function to be used to access a virtual directory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **callback** is not a valid pointer.

References VirtualDirCallbacks::open, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and virtualDirCallback.

7.1.5.87 UpnpVirtualDir_set_ReadCallback()

```
int UpnpVirtualDir_set_ReadCallback (
            VDCallback_Read callback )
```

Sets the read callback function to be used to access a virtual directory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **callback** is not a valid pointer.

References VirtualDirCallbacks::read, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and virtualDirCallback.

7.1.5.88 UpnpVirtualDir_set_SeekCallback()

```
int UpnpVirtualDir_set_SeekCallback (
            VDCallback_Seek callback )
```

Sets the seek callback function to be used to access a virtual directory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **callback** is not a valid pointer.

References VirtualDirCallbacks::seek, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and virtualDirCallback.

7.1.5.89 UpnpVirtualDir_set_WriteCallback()

```
int UpnpVirtualDir_set_WriteCallback (
            VDCallback_Write callback )
```

Sets the write callback function to be used to access a virtual directory.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_ARGUMENT`: **callback** is not a valid pointer.

References UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, virtualDirCallback, and VirtualDirCallbacks::write.

7.1.5.90 UpnpWriteHttpPost()

```
int UpnpWriteHttpPost (
             void * handle,
             char * buf,
             size_t * size,
             int timeout )
```

Sends a request to a server to copy the contents of a buffer to the URI specified in the **UpnpOpenHttpPost** call.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, **buf** or **size** is not a valid pointer.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*handle*	The handle of the connection created by the call to **UpnpOpenHttpPost**.
in	*buf*	The buffer to be posted.
in	*size*	The size, in bytes of **buf**.
in	*timeout*	A timeout value sent with the request during which a response is expected from the server, failing which, an error is reported. If value is negative, timeout is infinite.

References http_WriteHttpRequest().

7.1.5.91 UpnpWriteHttpRequest()

```
int UpnpWriteHttpRequest (
             void * handle,
             char * buf,
             size_t * size,
             int timeout )
```

Writes the content of a HTTP request initiated by a **UpnpMakeHttpRequest** call. The end of the content should be indicated by a call to **UpnpEndHttpRequest**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, **buf** or **size** is not a valid pointer.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*handle*	The handle of the connection created by the call to **UpnpOpenHttpConnection**.
in	*buf*	The buffer containing date to be written.
in	*size*	The size, in bytes of **buf**.
in	*timeout*	A timeout value sent with the request during which a response is expected from the server, failing which, an error is reported. If value is negative, timeout is infinite.

References http_WriteHttpRequest().

7.1.5.92 WinsockInit()

```
static int WinsockInit (
            void  )  [static]
```

(Windows Only) Initializes the Windows Winsock library.

Returns

UPNP_E_SUCCESS on success, UPNP_E_INIT_FAILED on failure.

References UPNP_E_INIT_FAILED, and UPNP_E_SUCCESS.

Referenced by UpnpInitPreamble().

7.1.6 Variable Documentation

7.1.6.1 bWebServerState

```
WebServerState bWebServerState = WEB_SERVER_DISABLED
```

Flag to indicate the state of web server

Referenced by UpnpEnableWebserver(), UpnpIsWebserverEnabled(), web_server_destroy(), and web_server_init().

7.1.6.2 g_maxContentLength

```
size_t g_maxContentLength = DEFAULT_SOAP_CONTENT_LENGTH
```

Maximum content-length (in bytes) that the SDK will process on an incoming packet. Content-Length exceeding this size will be not processed and error 413 (HTTP Error Code) will be returned to the remote end point.

Referenced by http_RecvMessage(), UpnpSetContentLength(), and UpnpSetMaxContentLength().

7.1.6.3 g_UpnpSdkEQMaxAge

```
int g_UpnpSdkEQMaxAge = MAX_SUBSCRIPTION_EVENT_AGE
```

Global variable to determine the maximum number of seconds which an event can spend on a subscription queue (waiting for the event at the head of the queue to be communicated). This parameter will have no effect in most situations with the default (low) value of MAX_SUBSCRIPTION_QUEUED_EVENTS. However, if MAX_↩ SUBSCRIPTION_QUEUED_EVENTS is set to a high value, the AGE parameter will allow pruning the queue in good conformance with the UPnP Device Architecture standard, at the price of higher potential memory use.

7.1.6.4 g_UpnpSdkEQMaxLen

```
int g_UpnpSdkEQMaxLen = MAX_SUBSCRIPTION_QUEUED_EVENTS
```

Global variable to determines the maximum number of events which can be queued for a given subscription before events begin to be discarded. This limits the amount of memory used for a non-responding subscribed entity.

7.1.6.5 gAllowLiteralHostRedirection

```
int gAllowLiteralHostRedirection = 0
```

Allow literal host names redirection to numeric host names.

7.1.6.6 gDocumentRootDir

```
membuffer gDocumentRootDir  [extern]
```

a local dir which serves as webserver root

Global variable. A local dir which serves as webserver root.

Referenced by process_request(), UpnpInitStartServers(), web_server_destroy(), web_server_init(), and web_server_set_root_dir().

7.1.6.7 gIF_INDEX

```
unsigned gIF_INDEX = (unsigned)-1
```

Contains interface index. (extern'ed in upnp.h)

Referenced by DeviceAdvertisement(), DeviceShutdown(), NewRequestHandler(), parse_hostport(), SearchByTarget(), ServiceAdvertisement(), ServiceShutdown(), and UpnpGetIfInfo().

7.1.6.8 gIF_IPV4

```
char gIF_IPV4[INET_ADDRSTRLEN] = {'\0'}
```

Static buffer to contain interface IPv4 address. (extern'ed in upnp.h)

Referenced by create_ssdp_sock_v4(), gena_subscribe(), gena_validate_delivery_urls(), get_ssdp_sockets(), NewRequestHandler(), SearchByTarget(), UpnpGetIfInfo(), and UpnpGetServerIpAddress().

7.1.6.9 gIF_IPV4_NETMASK

```
char gIF_IPV4_NETMASK[INET_ADDRSTRLEN] = {'\0'}
```

Static buffer to contain interface IPv4 netmask. (extern'ed in upnp.h)

Referenced by gena_validate_delivery_urls(), and UpnpGetIfInfo().

7.1.6.10 gIF_IPV6

```
char gIF_IPV6[INET6_ADDRSTRLEN] = {'\0'}
```

Static buffer to contain interface IPv6 link-local address (LLA). (extern'ed in upnp.h)

Referenced by gena_subscribe(), gena_validate_delivery_urls(), get_ssdp_sockets(), UpnpGetIfInfo(), and UpnpGetServerIp6Address().

7.1.6.11 gIF_IPV6_PREFIX_LENGTH

```
unsigned gIF_IPV6_PREFIX_LENGTH = 0
```

IPv6 LLA prefix length. (extern'ed in upnp.h)

Referenced by gena_validate_delivery_urls(), and UpnpGetIfInfo().

7.1.6.12 gIF_IPV6_ULA_GUA

```
char gIF_IPV6_ULA_GUA[INET6_ADDRSTRLEN] = {'\0'}
```

Static buffer to contain interface IPv6 unique-local or globally-unique address (ULA or GUA). (extern'ed in upnp.h)

Referenced by gena_subscribe(), gena_validate_delivery_urls(), get_ssdp_sockets(), UpnpGetIfInfo(), and UpnpGetServerUlaGuaIp6Address().

7.1.6.13 gIF_IPV6_ULA_GUA_PREFIX_LENGTH

```
unsigned gIF_IPV6_ULA_GUA_PREFIX_LENGTH = 0
```

IPv6 ULA or GUA prefix length. (extern'ed in upnp.h)

Referenced by gena_validate_delivery_urls(), and UpnpGetIfInfo().

7.1.6.14 gIF_NAME

```
char gIF_NAME[LINE_SIZE] = {'\0'}
```

Static buffer to contain interface name. (extern'ed in upnp.h)

Referenced by UpnpGetIfInfo().

7.1.6.15 GlobalClientSubscribeMutex

```
ithread_mutex_t GlobalClientSubscribeMutex
```

Mutex to synchronize the subscription handling at the client side.

Referenced by UpnpFinish(), and UpnpInitMutexes().

7.1.6.16 GlobalHndRWLock

```
ithread_rwlock_t GlobalHndRWLock
```

rwlock to synchronize handles (root device or control point handle).

Referenced by UpnpFinish(), and UpnpInitMutexes().

7.1.6.17 gMiniServerThreadPool

```
ThreadPool gMiniServerThreadPool
```

Mini server thread pool.

Referenced by StartMiniServer(), UpnpFinish(), and UpnpInitThreadPools().

7.1.6.18 gRecvThreadPool

```
ThreadPool gRecvThreadPool
```

Receive thread pool.

Referenced by readFromSSDPSocket(), ssdp_handle_ctrlpt_msg(), UpnpFinish(), and UpnpInitThreadPools().

7.1.6.19 gSDKInitMutex

```
ithread_mutex_t gSDKInitMutex = PTHREAD_MUTEX_INITIALIZER
```

Initialization mutex.

Referenced by UpnpInit2().

7.1.6.20 gSendThreadPool

```
ThreadPool gSendThreadPool
```

Send thread pool.

Referenced by genaNotifyThread(), UpnpFinish(), UpnpGetServiceVarStatusAsync(), UpnpInitPreamble(), UpnpInitThreadPools(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

7.1.6.21 gTimerThread

```
TimerThread gTimerThread
```

Global timer thread.

Referenced by free_client_subscription(), genaRenewSubscription(), ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_device_request(), UpnpFinish(), UpnpInitPreamble(), and UpnpSendAdvertisementLowPower().

7.1.6.22 gUUIDMutex

```
ithread_mutex_t gUUIDMutex
```

Mutex to synchronize the uuid creation process.

Referenced by UpnpFinish(), and UpnpInitMutexes().

7.1.6.23 gWebCallback_HostValidate

```
WebCallback_HostValidate gWebCallback_HostValidate = 0
```

webCallback for HOST validation.

7.1.6.24 gWebCallback_HostValidateCookie

```
void* gWebCallback_HostValidateCookie = 0
```

Cookie to the webCallback for HOST validation.

7.1.6.25 gWebserverCorsString

```
membuffer gWebserverCorsString  [extern]
```

a string which is set in the header field

Global variable. A string which is set in the header field.

Referenced by process_request(), UpnpInitStartServers(), web_server_destroy(), web_server_init(), and web_server_set_cors().

7.1.6.26 HandleTable

```
void* HandleTable[NUM_HANDLE]  [static]
```

UPnP device and control point handle table

Referenced by FreeHandle(), GetCallBackFn(), GetFreeHandle(), GetHandleInfo(), PrintHandleInfo(), UpnpInitPreamble(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

7.1.6.27 LOCAL_PORT_V4

```
unsigned short LOCAL_PORT_V4
```

local IPv4 port for the mini-server

Referenced by gena_subscribe(), UpnpGetServerPort(), and UpnpInitStartServers().

7.1.6.28 LOCAL_PORT_V6

```
unsigned short LOCAL_PORT_V6
```

IPv6 LLA port for the mini-server

Referenced by gena_subscribe(), UpnpGetServerPort6(), and UpnpInitStartServers().

7.1.6.29 LOCAL_PORT_V6_ULA_GUA

```
unsigned short LOCAL_PORT_V6_ULA_GUA
```

IPv6 ULA or GUA port for the mini-server

Referenced by gena_subscribe(), UpnpGetServerUlaGuaPort6(), and UpnpInitStartServers().

7.1.6.30 pVirtualDirList

```
virtualDirList* pVirtualDirList
```

Pointer to the virtual directory list.

Referenced by isFileInVirtualDir(), UpnpAddVirtualDir(), UpnpRemoveAllVirtualDirs(), UpnpRemoveVirtualDir(), and web_server_init().

7.1.6.31 UpnpSdkClientRegistered

```
int UpnpSdkClientRegistered = 0
```

Global variable to denote the state of Upnp SDK client registration. == 0 if unregistered, >= 1 if registered - registered clients count.

Referenced by UpnpRegisterClient(), and UpnpUnRegisterClient().

7.1.6.32 UpnpSdkDeviceRegisteredV4

```
int UpnpSdkDeviceRegisteredV4 = 0
```

Global variable to denote the state of Upnp SDK IPv4 device registration. == 0 if unregistered, == 1 if registered.

Referenced by GetDeviceHandleInfo(), GetDeviceHandleInfoForPath(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpUnRegisterRootDeviceLowPower().

7.1.6.33 UpnpSdkDeviceregisteredV6

```
int UpnpSdkDeviceregisteredV6 = 0
```

Global variable to denote the state of Upnp SDK IPv6 device registration. == 0 if unregistered, == 1 if registered.

Referenced by GetDeviceHandleInfo(), GetDeviceHandleInfoForPath(), UpnpRegisterClient(), UpnpRegisterRootDevice4(), and UpnpUnRegisterRootDeviceLowPower().

7.1.6.34 UpnpSdkInit

```
int UpnpSdkInit = 0
```

Global variable to denote the state of Upnp SDK == 0 if uninitialized, == 1 if initialized.

Referenced by UpnpAcceptSubscription(), UpnpAcceptSubscriptionExt(), UpnpAddVirtualDir(), UpnpEnableWebserver(), UpnpFinish(), UpnpGetServerIp6Address(), UpnpGetServerIpAddress(), UpnpGetServerPort(), UpnpGetServerPort6(), UpnpGetServerUlaGuaIp6Address(), UpnpGetServerUlaGuaPort6(), UpnpGetServiceVarStatus(), UpnpGetServiceVarStatusAsync(), UpnpInit2(), UpnpInitThreadPools(), UpnpIsWebserverEnabled(), UpnpNotify(), UpnpNotifyExt(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), UpnpRemoveAllVirtualDirs(), UpnpRemoveVirtualDir(), UpnpRenewSubscription(), UpnpRenewSubscriptionAsync(), UpnpSearchAsync(), UpnpSendAction(), UpnpSendActionAsync(), UpnpSendActionEx(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSetContentLength(), UpnpSetMaxContentLength(), UpnpSetMaxSubscriptions(), UpnpSetMaxSubscriptionTimeOut(), UpnpSubscribe(), UpnpSubscribeAsync(), UpnpUnRegisterClient(), UpnpUnRegisterRootDeviceLowPower(), UpnpUnSubscribe(), and UpnpUnSubscribeAsync().

7.1.6.35 virtualDirCallback

```
struct VirtualDirCallbacks virtualDirCallback
```

This structure is for virtual directory callbacks

Referenced by http_RecvPostMessage(), http_SendMessage(), process_request(), UpnpVirtualDir_set_CloseCallback(), UpnpVirtualDir_set_GetInfoCallback(), UpnpVirtualDir_set_OpenCallback(), UpnpVirtualDir_set_ReadCallback(), UpnpVirtualDir_set_SeekCallback(), UpnpVirtualDir_set_WriteCallback(), and web_server_init().

7.2 The UpnpString Class

Implements string operations in the UPnP library.

Files

- file UpnpString.h

  UpnpString object declaration.
- file UpnpString.c

  UpnpString object implementation.

Data Structures

- struct SUpnpString

  Internal implementation of the class UpnpString.

Typedefs

- typedef struct s_UpnpString UpnpString

  Type of the string objects inside libupnp.

Functions

- UpnpString * UpnpString_new (void)

  Constructor.
- void UpnpString_delete (UpnpString *p)

  Destructor.
- UpnpString * UpnpString_dup (const UpnpString *p)

  Copy Constructor.
- void UpnpString_assign (UpnpString *p, const UpnpString *q)

  Assignment operator.
- size_t UpnpString_get_Length (const UpnpString *p)

  Returns the length of the string.
- void UpnpString_set_Length (UpnpString *p, size_t n)

  Truncates the string to the specified lenght, or does nothing if the current lenght is less than or equal to the requested length.
- const char * UpnpString_get_String (const UpnpString *p)

  Returns the pointer to char.
- int UpnpString_set_String (UpnpString *p, const char *s)

  Sets the string from a pointer to char.
- int UpnpString_set_StringN (UpnpString *p, const char *s, size_t n)

  Sets the string from a pointer to char using a maximum of N chars.
- void UpnpString_clear (UpnpString *p)

  Clears the string, sets its size to zero.
- int UpnpString_cmp (UpnpString *p, UpnpString *q)

  Compares two strings for equality. Case matters.
- int UpnpString_casecmp (UpnpString *p, UpnpString *q)

  Compares two strings for equality. Case does not matter.
- static size_t **strnlen** (const char *s, size_t n)
- static char * **strndup** (const char *__string, size_t __n)

Variables

- size_t SUpnpString::m_length

  Length of the string.
- char * SUpnpString::m_string

  Pointer to a dynamically allocated area that holds the NULL terminated string.

7.2.1 Detailed Description

Implements string operations in the UPnP library.

Author

Marcelo Roberto Jimenez

Version

1.0

Due to its heavy use, this class is coded for efficiency, not for beauty. Do not use this as example to other classes. Please take a look at any other one.

Todo Always alloc a minimum size like 64 bytes or so and when shrinking do not perform a new memory allocation.

7.2.2 Typedef Documentation

7.2.2.1 UpnpString

```
typedef struct s_UpnpString UpnpString
```

Type of the string objects inside libupnp.

7.2.3 Function Documentation

7.2.3.1 UpnpString_assign()

```
void UpnpString_assign (
            UpnpString * p,
            const UpnpString * q )
```

Assignment operator.

Parameters

in	*p*	The ***this*** *pointer.*
in	*q*	The ***that*** *pointer.*

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by genaSubscribe().

7.2.3.2 UpnpString_casecmp()

```
int UpnpString_casecmp (
```

```
            UpnpString * p,
            UpnpString * q )
```

Compares two strings for equality. Case does not matter.

Returns

The result of strcasecmp().

Parameters

in	*p*	The ***the*** *first* string.
in	*q*	The ***the*** *second* string.

References UpnpString_get_String().

7.2.3.3 UpnpString_clear()

```
void UpnpString_clear (
            UpnpString * p )
```

Clears the string, sets its size to zero.

Parameters

in	*p*	The ***this*** *pointer.*

Referenced by gena_subscribe(), genaSubscribe(), GenlibClientSubscription_clear_ActualSID(), GenlibClientSubscription_clear_Eve GenlibClientSubscription_clear_SID(), TestClass_clear_TheString(), UpnpActionComplete_clear_CtrlUrl(), UpnpActionRequest_clear_ActionName(), UpnpActionRequest_clear_DevUDN(), UpnpActionRequest_clear_ErrStr(), UpnpActionRequest_clear_Os(), UpnpActionRequest_clear_ServiceID(), UpnpDiscovery_clear_Date(), UpnpDiscovery_clear_Device UpnpDiscovery_clear_DeviceType(), UpnpDiscovery_clear_Ext(), UpnpDiscovery_clear_Location(), UpnpDiscovery_clear_Os(), UpnpDiscovery_clear_ServiceType(), UpnpDiscovery_clear_ServiceVer(), UpnpEvent_clear_SID(), UpnpEventSubscribe_clear_Publ UpnpEventSubscribe_clear_SID(), UpnpExtraHeaders_clear_name(), UpnpExtraHeaders_clear_value(), UpnpFileInfo_clear_Os(), UpnpStateVarComplete_clear_CtrlUrl(), UpnpStateVarComplete_clear_StateVarName(), UpnpStateVarRequest_clear_DevUDN(), UpnpStateVarRequest_clear_ErrStr(), UpnpStateVarRequest_clear_ServiceID(), UpnpStateVarRequest_clear_StateVarName(), UpnpSubscriptionRequest_clear_ServiceId(), UpnpSubscriptionRequest_clear_SID(), and UpnpSubscriptionRequest_clear_UDN().

7.2.3.4 UpnpString_cmp()

```
int UpnpString_cmp (
            UpnpString * p,
            UpnpString * q )
```

Compares two strings for equality. Case matters.

Returns

The result of strcmp().

Parameters

in	*p*	The ***the*** *first* string.
in	*q*	The ***the*** *second* string.

References UpnpString_get_String().

7.2.3.5 UpnpString_delete()

```
void UpnpString_delete (
            UpnpString * p )
```

Destructor.

Parameters

in	*p*	The ***this*** *pointer.*

References SUpnpString::m_length, and SUpnpString::m_string.

Referenced by genaRenewSubscription(), genaSubscribe(), GenlibClientSubscription_delete(), TestClass_delete(), UpnpActionComplete_delete(), UpnpActionRequest_delete(), UpnpDiscovery_delete(), UpnpEvent_delete(), UpnpEventSubscribe_delete(), UpnpExtraHeaders_delete(), UpnpFileInfo_delete(), UpnpRenewSubscription(), UpnpStateVarComplete_delete(), UpnpStateVarRequest_delete(), UpnpSubscribe(), UpnpSubscriptionRequest_delete(), UpnpThreadDistribution(), and UpnpUnSubscribe().

7.2.3.6 UpnpString_dup()

```
UpnpString * UpnpString_dup (
            const UpnpString * p )
```

Copy Constructor.

Returns

A pointer to a new allocated copy of the original object.

Parameters

in	*p*	The ***this*** *pointer.*

References SUpnpString::m_length, and SUpnpString::m_string.

7.2.3.7 UpnpString_get_Length()

```
size_t UpnpString_get_Length (
            const UpnpString * p )
```

Returns the length of the string.

Returns

The length of the string.

Parameters

in	*p*	The ***this*** *pointer.*

Referenced by gena_subscribe(), gena_unsubscribe(), GenlibClientSubscription_get_ActualSID_Length(), GenlibClientSubscription_get_EventURL_Length(), GenlibClientSubscription_get_SID_Length(), handle_query_variable(), ssdp_handle_ctrlpt_msg(), TestClass_get_TheString_Length(), UpnpActionComplete_get_CtrlUrl_Length(), UpnpActionRequest_get_ActionName_Length(), UpnpActionRequest_get_DevUDN_Length(), UpnpActionRequest_get_ErrStr_Leng UpnpActionRequest_get_Os_Length(), UpnpActionRequest_get_ServiceID_Length(), UpnpDiscovery_get_Date_Length(), UpnpDiscovery_get_DeviceID_Length(), UpnpDiscovery_get_DeviceType_Length(), UpnpDiscovery_get_Ext_Length(), UpnpDiscovery_get_Location_Length(), UpnpDiscovery_get_Os_Length(), UpnpDiscovery_get_ServiceType_Length(), UpnpDiscovery_get_ServiceVer_Length(), UpnpEvent_get_SID_Length(), UpnpEventSubscribe_get_PublisherUrl_Length(), UpnpEventSubscribe_get_SID_Length(), UpnpExtraHeaders_get_name_Length(), UpnpExtraHeaders_get_value_Length(), UpnpFileInfo_get_Os_Length(), UpnpStateVarComplete_get_CtrlUrl_Length(), UpnpStateVarComplete_get_StateVarName_Length() UpnpStateVarRequest_get_DevUDN_Length(), UpnpStateVarRequest_get_ErrStr_Length(), UpnpStateVarRequest_get_ServiceID_ UpnpStateVarRequest_get_StateVarName_Length(), UpnpSubscriptionRequest_get_ServiceId_Length(), UpnpSubscriptionRequest and UpnpSubscriptionRequest_get_UDN_Length().

7.2.3.8 UpnpString_get_String()

```
const char * UpnpString_get_String (
            const UpnpString * p )
```

Returns the pointer to char.

Returns

The pointer to char.

Parameters

in	*p*	The ***this*** *pointer.*

Referenced by gena_subscribe(), gena_unsubscribe(), GenlibClientSubscription_get_ActualSID_cstr(), GenlibClientSubscription_get GenlibClientSubscription_get_SID_cstr(), GenlibClientSubscription_set_ActualSID(), GenlibClientSubscription_set_EventURL(), GenlibClientSubscription_set_SID(), GetClientSubClientSID(), RemoveClientSubClientSID(), SampleUtil_PrintEvent(), TestClass_get_TheString_cstr(), TestClass_set_TheString(), TvDeviceHandleActionRequest(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), UpnpActionComplete_get_CtrlUrl_cstr(), UpnpActionComplete_set_CtrlUrl(), UpnpActionRequest_get_ActionName_cstr(), UpnpActionRequest_get_DevUDN_cstr(), UpnpActionRequest_get_ErrStr_cstr(), UpnpActionRequest_get_Os_cstr(), UpnpActionRequest_get_ServiceID_cstr(), UpnpActionRequest_set_ActionName(), UpnpActionRequest_set_DevUDN(), UpnpActionRequest_set_ErrStr(), UpnpActionRequest_set_Os(), UpnpActionRequest_set_Serv UpnpDiscovery_get_Date_cstr(), UpnpDiscovery_get_DeviceID_cstr(), UpnpDiscovery_get_DeviceType_cstr(), UpnpDiscovery_get_Ext_cstr(), UpnpDiscovery_get_Location_cstr(), UpnpDiscovery_get_Os_cstr(), UpnpDiscovery_get_ServiceTyp UpnpDiscovery_get_ServiceVer_cstr(), UpnpDiscovery_set_Date(), UpnpDiscovery_set_DeviceID(), UpnpDiscovery_set_DeviceType UpnpDiscovery_set_Ext(), UpnpDiscovery_set_Location(), UpnpDiscovery_set_Os(), UpnpDiscovery_set_ServiceType(), UpnpDiscovery_set_ServiceVer(), UpnpEvent_get_SID_cstr(), UpnpEvent_set_SID(), UpnpEventSubscribe_get_PublisherUrl_cstr(), UpnpEventSubscribe_get_SID_cstr(), UpnpEventSubscribe_set_PublisherUrl(), UpnpEventSubscribe_set_SID(), UpnpExtraHeaders_get_name_cstr(), UpnpExtraHeaders_get_value_cstr(), UpnpExtraHeaders_set_name(), UpnpExtraHeaders_set_value(), UpnpFileInfo_get_Os_cstr(), UpnpFileInfo_set_Os(), UpnpStateVarComplete_get_CtrlUrl_cstr(),

UpnpStateVarComplete_get_StateVarName_cstr(), UpnpStateVarComplete_set_CtrlUrl(), UpnpStateVarComplete_set_StateVarNam
UpnpStateVarRequest_get_DevUDN_cstr(), UpnpStateVarRequest_get_ErrStr_cstr(), UpnpStateVarRequest_get_ServiceID_cstr(),
UpnpStateVarRequest_get_StateVarName_cstr(), UpnpStateVarRequest_set_DevUDN(), UpnpStateVarRequest_set_ErrStr(),
UpnpStateVarRequest_set_ServiceID(), UpnpStateVarRequest_set_StateVarName(), UpnpString_assign(),
UpnpString_casecmp(), UpnpString_cmp(), UpnpSubscribe(), UpnpSubscriptionRequest_get_ServiceId_cstr(),
UpnpSubscriptionRequest_get_SID_cstr(), UpnpSubscriptionRequest_get_UDN_cstr(), UpnpSubscriptionRequest_set_ServiceId(),
UpnpSubscriptionRequest_set_SID(), and UpnpSubscriptionRequest_set_UDN().

7.2.3.9 UpnpString_new()

```
UpnpString * UpnpString_new (
             void  )
```

Constructor.

Returns

A pointer to a new allocated object.

References SUpnpString::m_length, and SUpnpString::m_string.

Referenced by genaRenewSubscription(), genaSubscribe(), GenlibClientSubscription_new(), TestClass_new(), UpnpActionComplete_new(), UpnpActionRequest_new(), UpnpDiscovery_new(), UpnpEvent_new(), UpnpEventSubscribe_new(), UpnpExtraHeaders_new(), UpnpFileInfo_new(), UpnpRenewSubscription(), UpnpStateVarComplete_new(), UpnpStateVarRequest_new(), UpnpSubscribe(), UpnpSubscriptionRequest_new(), UpnpThreadDistribution(), and UpnpUnSubscribe().

7.2.3.10 UpnpString_set_Length()

```
void UpnpString_set_Length (
             UpnpString * p,
             size_t n )
```

Truncates the string to the specified lenght, or does nothing if the current lenght is less than or equal to the requested length.

Parameters

in	*p*	The ***this*** *pointer.*
in	*n*	The requested length.

References SUpnpString::m_length.

7.2.3.11 UpnpString_set_String()

```
int UpnpString_set_String (
             UpnpString * p,
             const char * s )
```

Sets the string from a pointer to char.

Parameters

in	*p*	The ***this*** *pointer.*
in	*s*	(char *) to copy from.

References SUpnpString::m_string.

Referenced by genaSubscribe(), GenlibClientSubscription_set_ActualSID(), GenlibClientSubscription_set_EventURL(), GenlibClientSubscription_set_SID(), GenlibClientSubscription_strcpy_ActualSID(), GenlibClientSubscription_strcpy_EventURL(), GenlibClientSubscription_strcpy_SID(), TestClass_set_TheString(), TestClass_strcpy_TheString(), UpnpActionComplete_set_CtrlUrl(UpnpActionComplete_strcpy_CtrlUrl(), UpnpActionRequest_set_ActionName(), UpnpActionRequest_set_DevUDN(), UpnpActionRequest_set_ErrStr(), UpnpActionRequest_set_Os(), UpnpActionRequest_set_ServiceID(), UpnpActionRequest_strcpy_ UpnpActionRequest_strcpy_DevUDN(), UpnpActionRequest_strcpy_ErrStr(), UpnpActionRequest_strcpy_Os(), UpnpActionRequest_strcpy_ServiceID(), UpnpDiscovery_set_Date(), UpnpDiscovery_set_DeviceID(), UpnpDiscovery_set_DeviceTyp UpnpDiscovery_set_Ext(), UpnpDiscovery_set_Location(), UpnpDiscovery_set_Os(), UpnpDiscovery_set_ServiceType(), UpnpDiscovery_set_ServiceVer(), UpnpDiscovery_strcpy_Date(), UpnpDiscovery_strcpy_DeviceID(), UpnpDiscovery_strcpy_Device UpnpDiscovery_strcpy_Ext(), UpnpDiscovery_strcpy_Location(), UpnpDiscovery_strcpy_Os(), UpnpDiscovery_strcpy_ServiceType(), UpnpDiscovery_strcpy_ServiceVer(), UpnpEvent_set_SID(), UpnpEvent_strcpy_SID(), UpnpEventSubscribe_set_PublisherUrl(), UpnpEventSubscribe_set_SID(), UpnpEventSubscribe_strcpy_PublisherUrl(), UpnpEventSubscribe_strcpy_SID(), UpnpExtraHeaders_set_name(), UpnpExtraHeaders_set_value(), UpnpExtraHeaders_strcpy_name(), UpnpExtraHeaders_strcpy_val UpnpFileInfo_set_Os(), UpnpFileInfo_strcpy_Os(), UpnpRenewSubscription(), UpnpStateVarComplete_set_CtrlUrl(), UpnpStateVarComplete_set_StateVarName(), UpnpStateVarComplete_strcpy_CtrlUrl(), UpnpStateVarComplete_strcpy_StateVarNar UpnpStateVarRequest_set_DevUDN(), UpnpStateVarRequest_set_ErrStr(), UpnpStateVarRequest_set_ServiceID(), UpnpStateVarRequest_set_StateVarName(), UpnpStateVarRequest_strcpy_DevUDN(), UpnpStateVarRequest_strcpy_ErrStr(), UpnpStateVarRequest_strcpy_ServiceID(), UpnpStateVarRequest_strcpy_StateVarName(), UpnpString_assign(), UpnpSubscribe(), UpnpSubscriptionRequest_set_ServiceId(), UpnpSubscriptionRequest_set_SID(), UpnpSubscriptionRequest_set_ UpnpSubscriptionRequest_strcpy_ServiceId(), UpnpSubscriptionRequest_strcpy_SID(), UpnpSubscriptionRequest_strcpy_UDN(), and UpnpUnSubscribe().

7.2.3.12 UpnpString_set_StringN()

```
int UpnpString_set_StringN (
            UpnpString * p,
            const char * s,
            size_t n )
```

Sets the string from a pointer to char using a maximum of N chars.

Parameters

in	*p*	The ***this*** *pointer.*
in	*s*	(char *) to copy from.
	n	Maximum number of chars to copy.

References SUpnpString::m_string.

Referenced by gena_subscribe(), GenlibClientSubscription_strncpy_ActualSID(), GenlibClientSubscription_strncpy_EventURL(), GenlibClientSubscription_strncpy_SID(), TestClass_strncpy_TheString(), UpnpActionComplete_strncpy_CtrlUrl(), UpnpActionRequest_strncpy_ActionName(), UpnpActionRequest_strncpy_DevUDN(), UpnpActionRequest_strncpy_ErrStr(), UpnpActionRequest_strncpy_Os(), UpnpActionRequest_strncpy_ServiceID(), UpnpDiscovery_strncpy_Date(), UpnpDiscovery_strncpy_DeviceID(), UpnpDiscovery_strncpy_DeviceType(), UpnpDiscovery_strncpy_Ext(), UpnpDiscovery_strncpy_Location(), UpnpDiscovery_strncpy_Os(), UpnpDiscovery_strncpy_ServiceType(), UpnpDiscovery_strncpy_ServiceVer(), UpnpEvent_strncpy_SID(), UpnpEventSubscribe_strncpy_PublisherUrl(),

UpnpEventSubscribe_strncpy_SID(), UpnpExtraHeaders_strncpy_name(), UpnpExtraHeaders_strncpy_value(), UpnpFileInfo_strncpy_Os(), UpnpStateVarComplete_strncpy_CtrlUrl(), UpnpStateVarComplete_strncpy_StateVarName(), UpnpStateVarRequest_strncpy_DevUDN(), UpnpStateVarRequest_strncpy_ErrStr(), UpnpStateVarRequest_strncpy_ServiceID(), UpnpStateVarRequest_strncpy_StateVarName(), UpnpSubscriptionRequest_strncpy_ServiceId(), UpnpSubscriptionRequest_strncpy and UpnpSubscriptionRequest_strncpy_UDN().

7.2.4 Variable Documentation

7.2.4.1 m_length

```
size_t SUpnpString::m_length
```

Length of the string.

Referenced by UpnpString_delete(), UpnpString_dup(), UpnpString_new(), and UpnpString_set_Length().

7.2.4.2 m_string

```
char* SUpnpString::m_string
```

Pointer to a dynamically allocated area that holds the NULL terminated string.

Referenced by UpnpString_delete(), UpnpString_dup(), UpnpString_new(), UpnpString_set_String(), and UpnpString_set_StringN().

7.3 Optional Tool API

Additional, optional utility API that can be helpful in writing applications.

Functions

- UPNP_EXPORT_SPEC void UpnpSetMaxJobsTotal (int mjt)

  Sets the maximum number of jobs in the internal thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.
- UPNP_EXPORT_SPEC const char * UpnpGetErrorMessage (int errorcode)

  Converts an SDK error code into a string error message suitable for display. The memory returned from this function should NOT be freed.
- UPNP_EXPORT_SPEC int UpnpResolveURL (const char *BaseURL, const char *RelURL, char *AbsURL)

  Combines a base URL and a relative URL into a single absolute URL.
- UPNP_EXPORT_SPEC int UpnpResolveURL2 (const char *BaseURL, const char *RelURL, char **Abs↩ URL)

  Combines a base URL and a relative URL into a single absolute URL.
- UPNP_EXPORT_SPEC IXML_Document * UpnpMakeAction (const char *ActionName, const char *Serv↩ Type, int NumArg, const char *Arg,...)

  Creates an action request packet based on its input parameters (status variable name and value pair).
- UPNP_EXPORT_SPEC IXML_Document * UpnpMakeActionResponse (const char *ActionName, const char *ServType, int NumArg, const char *Arg,...)

  Ceates an action response packet based on its output parameters (status variable name and value pair).

- UPNP_EXPORT_SPEC int UpnpAddToAction (IXML_Document **ActionDoc, const char *ActionName, const char *ServType, const char *ArgName, const char *ArgVal)

  Adds the argument in the action request.
- UPNP_EXPORT_SPEC int UpnpAddToActionResponse (IXML_Document **ActionResponse, const char *ActionName, const char *ServType, const char *ArgName, const char *ArgVal)

  Creates an action response packet based on its output parameters (status variable name and value pair).
- UPNP_EXPORT_SPEC IXML_Document * UpnpCreatePropertySet (int NumArg, const char *Arg,...)

  Creates a property set message packet.
- UPNP_EXPORT_SPEC int UpnpAddToPropertySet (IXML_Document **PropSet, const char *ArgName, const char *ArgVal)

  Can be used when an application needs to transfer the status of many variables at once.

7.3.1 Detailed Description

Additional, optional utility API that can be helpful in writing applications.

This additional API can be compiled out in order to save code size in the library. Refer to the file README for details.

7.3.2 Function Documentation

7.3.2.1 UpnpAddToAction()

```
UPNP_EXPORT_SPEC int UpnpAddToAction (
             IXML_Document ** ActionDoc,
             const char * ActionName,
             const char * ServType,
             const char * ArgName,
             const char * ArgVal )
```

Adds the argument in the action request.

This API is specially suitable inside a loop to add any number input parameters into an existing action. If no action document exists in the beginning then a **Upnp_Document variable initialized with `NULL`** should be passed as a parameter.

It is a wrapper function that calls addToAction() function to add the argument in the action request.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: One or more of the parameters are invalid.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

`in, out`	*ActionDoc*	A pointer to store the action document node.
`in`	*ActionName*	The action name.
`in`	*ServType*	The service type.
`in`	*ArgName*	The status variable name.
`in`	*ArgVal*	The status variable value.

References addToAction().

7.3.2.2 UpnpAddToActionResponse()

```
UPNP_EXPORT_SPEC int UpnpAddToActionResponse (
             IXML_Document ** ActionResponse,
             const char * ActionName,
             const char * ServType,
             const char * ArgName,
             const char * ArgVal )
```

Creates an action response packet based on its output parameters (status variable name and value pair).

This API is especially suitable inside a loop to add any number of input parameters into an existing action response. If no action document exists in the beginning, a **Upnp_Document** variable initialized with NULL should be passed as a parameter.

It is a wrapper function that calls addToAction() function to add the argument in the action request.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: One or more of the parameters are invalid.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Parameters

in,out	*ActionResponse*	Pointer to a document to store the action document node.
in	*ActionName*	The action name.
in	*ServType*	The service type.
in	*ArgName*	The status variable name.
in	*ArgVal*	The status variable value.

References addToAction().

Referenced by IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), TvDevicePowerOff(), TvDevicePowerOn(), TvDeviceSetBrightness(), TvDeviceSetChannel(), TvDeviceSetColor(), TvDeviceSetContrast(), TvDeviceSetTint(), and TvDeviceSetVolume().

7.3.2.3 UpnpAddToPropertySet()

```
UPNP_EXPORT_SPEC int UpnpAddToPropertySet (
             IXML_Document ** PropSet,
             const char * ArgName,
             const char * ArgVal )
```

Can be used when an application needs to transfer the status of many variables at once.

It can be used (inside a loop) to add some extra status variables into an existing property set. If the application does not already have a property set document, the application should create a variable initialized with NULL and pass that as the first parameter.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: One or more of the parameters are invalid.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to complete this operation.

Parameters

`in,out`	*PropSet*	A pointer to the document containing the property set document node.
`in`	*ArgName*	The status variable name.
`in`	*ArgVal*	The status variable value.

References ixmlDocument_createElement(), ixmlDocument_createTextNode(), ixmlNode_appendChild(), ixmlNode_getFirstChild(), ixmlParseBufferEx(), ErrorString::rc, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by TvDeviceHandleSubscriptionRequest().

7.3.2.4 UpnpCreatePropertySet()

```
UPNP_EXPORT_SPEC IXML_Document * UpnpCreatePropertySet (
             int NumArg,
             const char * Arg,
              ...  )
```

Creates a property set message packet.

Any number of input parameters can be passed to this function but every input variable name should have a matching value input argument.

Returns

`NULL` on failure, or the property-set document node.

Parameters

`in`	*NumArg*	The number of argument pairs passed.
`in`	*Arg*	The status variable name and value pair.
`in`	...	Variable sized list with the rest of the parameters.

References ixmlDocument_createElement(), ixmlDocument_createTextNode(), ixmlNode_appendChild(), ixmlNode_getFirstChild(), and ixmlParseBufferEx().

Referenced by TvDeviceSetServiceTableVar().

7.3.2.5 UpnpGetErrorMessage()

```
UPNP_EXPORT_SPEC const char * UpnpGetErrorMessage (
             int errorcode )
```

Converts an SDK error code into a string error message suitable for display. The memory returned from this function should NOT be freed.

Returns

An ASCII text string representation of the error message associated with the error code or the string "Unknown error code"

Parameters

in	*errorcode*	The SDK error code to convert.

References ErrorMessages, ErrorString::rc, and ErrorString::rcError.

7.3.2.6 UpnpMakeAction()

```
UPNP_EXPORT_SPEC IXML_Document * UpnpMakeAction (
             const char * ActionName,
             const char * ServType,
             int NumArg,
             const char * Arg,
              ... )
```

Creates an action request packet based on its input parameters (status variable name and value pair).

Any number of input parameters can be passed to this function but every input variable name should have a matching value argument.

It is a wrapper function that calls makeAction() function to create the action request.

Returns

The action node of **Upnp_Document** type or `NULL` if the operation failed.

Parameters

in	*ActionName*	Name of the action request or response.
in	*ServType*	The service type.
in	*NumArg*	Number of argument pairs to be passed.
in	*Arg*	pointer to the first argument.
in	...	Argument list.

References makeAction().

7.3.2.7 UpnpMakeActionResponse()

```
UPNP_EXPORT_SPEC IXML_Document * UpnpMakeActionResponse (
             const char * ActionName,
             const char * ServType,
```

```
            int NumArg,
            const char * Arg,
             ...  )
```

Ceates an action response packet based on its output parameters (status variable name and value pair).

Any number of input parameters can be passed to this function but every output variable name should have a matching value argument.

It is a wrapper function that calls makeAction() function to create the action request.

Returns

The action node of **Upnp_Document** type or NULL if the operation failed.

Parameters

in	*ActionName*	The action name.
in	*ServType*	The service type..
in	*NumArg*	The number of argument pairs passed.
in	*Arg*	The status variable name and value pair.
in	...	Other status variable name and value pairs.

References makeAction().

7.3.2.8 UpnpResolveURL()

```
UPNP_EXPORT_SPEC int UpnpResolveURL (
            const char * BaseURL,
            const char * RelURL,
            char * AbsURL )
```

Combines a base URL and a relative URL into a single absolute URL.

The memory for **AbsURL** needs to be allocated by the caller and must be large enough to hold the **BaseURL** and **RelURL** combined.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: **RelURL** is NULL.
- UPNP_E_INVALID_URL: The **BaseURL** / **RelURL** combination does not form a valid URL.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Todo There is some unnecessary allocation and deallocation going on here because of the way resolve_rel_url() was originally written and used. In the future it would be nice to clean this up.

Parameters

in	*BaseURL*	The base URL to combine.
in	*RelURL*	The relative URL to **BaseURL**.
out	*AbsURL*	A pointer to a buffer to store the absolute URL.

References resolve_rel_url(), UPNP_E_INVALID_PARAM, UPNP_E_INVALID_URL, and UPNP_E_SUCCESS.

7.3.2.9 UpnpResolveURL2()

```
UPNP_EXPORT_SPEC int UpnpResolveURL2 (
            const char * BaseURL,
            const char * RelURL,
            char ** AbsURL )
```

Combines a base URL and a relative URL into a single absolute URL.

The memory for **AbsURL** becomes owned by the caller and should be freed later.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: **RelURL** is NULL.
- UPNP_E_INVALID_URL: The **BaseURL** / **RelURL** combination does not form a valid URL.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to complete this operation.

Parameters

in	*BaseURL*	The base URL to combine.
in	*RelURL*	The relative URL to **BaseURL**.
out	*AbsURL*	A pointer to a pointer to a buffer to store the absolute URL. Must be freed later by the caller.

References resolve_rel_url(), UPNP_E_INVALID_PARAM, UPNP_E_INVALID_URL, and UPNP_E_SUCCESS.

Referenced by SampleUtil_FindAndParseService().

7.3.2.10 UpnpSetMaxJobsTotal()

```
UPNP_EXPORT_SPEC void UpnpSetMaxJobsTotal (
            int mjt )
```

Sets the maximum number of jobs in the internal thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.

References TPSetMaxJobsTotal().

7.4 Sample Code

Files

- file sample_util.c
- file sample_util.h
- file tv_ctrlpt.c
- file tv_ctrlpt.h
- file tv_device.c
- file tv_device.h

Data Structures

- struct cmdloop_commands
- struct tv_service
- struct TvDevice
- struct TvDeviceNode
- struct TvService

Typedefs

- typedef void(* print_string) (const char *string,...)

  Prototype for displaying strings. All printing done by the device, control point, and sample util, ultimately use this to display strings to the user.
- typedef void(* state_update) (const char *varName, const char *varValue, const char *UDN, eventType type)

  Prototype for passing back state changes.

Enumerations

- enum **eventType** { **STATE_UPDATE** = 0 , **DEVICE_ADDED** = 1 , **DEVICE_REMOVED** = 2 , **GET_VAR_↩ COMPLETE** = 3 }

Functions

- int SampleUtil_Initialize (print_string print_function)

  Initializes the sample util. Must be called before any sample util functions. May be called multiple times.
- int SampleUtil_RegisterUpdateFunction (state_update update_function)
- int SampleUtil_Finish ()

  Releases Resources held by sample util.
- char * SampleUtil_GetElementValue (IXML_Element *element)

  Given a DOM node such as <Channel>11</Channel>, this routine extracts the value (e.g., 11) from the node and returns it as a string. The string must be freed by the caller using free.
- IXML_NodeList * SampleUtil_GetFirstServiceList (IXML_Document *doc)

  Given a DOM node representing a UPnP Device Description Document, this routine parses the document and finds the first service list (i.e., the service list for the root device). The service list is returned as a DOM node list. The NodeList must be freed using NodeList_free.
- char * SampleUtil_GetFirstDocumentItem (IXML_Document *doc, const char *item)

  Given a document node, this routine searches for the first element named by the input string item, and returns its value as a string. String must be freed by caller using free.
- char * SampleUtil_GetFirstElementItem (IXML_Element *element, const char *item)

Given a DOM element, this routine searches for the first element named by the input string item, and returns its value as a string. The string must be freed using free.

- void SampleUtil_PrintEventType (Upnp_EventType S)

  Prints a callback event type as a string.
- int SampleUtil_PrintEvent (Upnp_EventType EventType, const void *Event)

  Prints callback event structure details.
- int SampleUtil_FindAndParseService (IXML_Document *DescDoc, const char *location, const char *serviceType, char **serviceId, char **eventURL, char **controlURL)

  This routine finds the first occurance of a service in a DOM representation of a description document and parses it. Note that this function currently assumes that the eventURL and controlURL values in the service definitions are full URLs. Relative URLs are not handled here.
- int SampleUtil_Print (const char *fmt, ...)

  Function emulating printf that ultimately calls the registered print function with the formatted string.
- void SampleUtil_StateUpdate (const char *varName, const char *varValue, const char *UDN, eventType type)
- void linux_print (const char *format,...)

  Prints a string to standard out.

Variables

- static int **initialize_init** = 1
- static int **initialize_register** = 1
- print_string gPrintFun = NULL
- state_update gStateUpdateFun = NULL
- ithread_mutex_t display_mutex
- const char * **TvServiceType** [ ]
- ithread_mutex_t display_mutex
- print_string gPrintFun
- state_update gStateUpdateFun

Control Point Sample Module

- enum cmdloop_tvcmds {
  PRTHELP = 0 , **PRTFULLHELP** , **POWON** , **POWOFF** ,
  SETCHAN , **SETVOL** , **SETCOL** , **SETTINT** ,
  SETCONT , **SETBRT** , **CTRLACTION** , **PICTACTION** ,
  CTRLGETVAR , **PICTGETVAR** , **PRTDEV** , **LSTDEV** ,
  REFRESH , **EXITCMD** }
- ithread_mutex_t DeviceListMutex
- UpnpClient_Handle **ctrlpt_handle** = -1
- const char TvDeviceType [ ] = "urn:schemas-upnp-org:device:tvdevice:1"
- const char * TvServiceName [ ] = {"Control", "Picture"}
- const char * TvVarName [2][4]
- char TvVarCount [2]
- int default_timeout = 1801
- struct TvDeviceNode * GlobalDeviceList = NULL
- static int TvCtrlPointTimerLoopRun = 1

  Function that runs in its own thread and monitors advertisement and subscription timeouts for devices in the global device list.
- struct cmdloop_commands **cmdloop_commands**
- static struct cmdloop_commands cmdloop_cmdlist [ ]
- int **TvCtrlPointDeleteNode** (struct TvDeviceNode *node)
- int **TvCtrlPointRemoveDevice** (const char *UDN)

- int **TvCtrlPointRemoveAll** (void)
- int **TvCtrlPointRefresh** (void)
- int **TvCtrlPointGetVar** (int service, int devnum, const char *varname)
- int **TvCtrlPointGetPower** (int devnum)
- int **TvCtrlPointGetChannel** (int devnum)
- int **TvCtrlPointGetVolume** (int devnum)
- int **TvCtrlPointGetColor** (int devnum)
- int **TvCtrlPointGetTint** (int devnum)
- int **TvCtrlPointGetContrast** (int devnum)
- int **TvCtrlPointGetBrightness** (int devnum)
- int **TvCtrlPointSendAction** (int service, int devnum, const char *actionname, const char **param_name, char **param_val, int param_count)
- int **TvCtrlPointSendActionNumericArg** (int devnum, int service, const char *actionName, const char *paramName, int paramValue)
- int **TvCtrlPointSendPowerOn** (int devnum)
- int **TvCtrlPointSendPowerOff** (int devnum)
- int **TvCtrlPointSendSetChannel** (int devnum, int channel)
- int **TvCtrlPointSendSetVolume** (int devnum, int volume)
- int **TvCtrlPointSendSetColor** (int devnum, int color)
- int **TvCtrlPointSendSetTint** (int devnum, int tint)
- int **TvCtrlPointSendSetContrast** (int devnum, int contrast)
- int **TvCtrlPointSendSetBrightness** (int devnum, int brightness)
- int **TvCtrlPointGetDevice** (int devnum, struct TvDeviceNode **devnode)
- int **TvCtrlPointPrintList** ()
- int **TvCtrlPointPrintDevice** (int devnum)
- void **TvCtrlPointAddDevice** (IXML_Document *DescDoc, const char *location, int expires)
- void TvStateUpdate (char *UDN, int Service, IXML_Document *ChangedVariables, char **State)

  Update a Tv state table. Called when an event is received.
- void **TvCtrlPointHandleEvent** (const char *sid, int evntkey, IXML_Document *changes)
- void **TvCtrlPointHandleSubscribeUpdate** (const char *eventURL, const Upnp_SID sid, int timeout)
- void **TvCtrlPointHandleGetVar** (const char *controlURL, const char *varName, const DOMString varValue)
- int **TvCtrlPointCallbackEventHandler** (Upnp_EventType EventType, const void *Event, void *Cookie)
- void TvCtrlPointVerifyTimeouts (int incr)

  Checks the advertisement each device in the global device list.
- void * **TvCtrlPointTimerLoop** (void *args)
- int TvCtrlPointStart (char *iface, state_update updateFunctionPtr, int combo)

  Call this function to initialize the UPnP library and start the TV Control Point. This function creates a timer thread and provides a callback handler to process any UPnP events that are received.
- int **TvCtrlPointStop** (void)
- void TvCtrlPointPrintShortHelp (void)

  Print help info for this application.
- void TvCtrlPointPrintLongHelp (void)

  Print long help info for this application.
- void TvCtrlPointPrintCommands (void)
- void * TvCtrlPointCommandLoop (void *args)

  Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands.
- int **TvCtrlPointProcessCommand** (char *cmdline)

Device Sample API

- typedef int(∗ upnp_action) (IXML_Document ∗request, IXML_Document ∗∗out, const char ∗∗errorString)

  Prototype for all actions. For each action that a service implements, there is a corresponding function with this prototype.
- struct TvService tv_service_table [ ]
- UpnpDevice_Handle device_handle
- ithread_mutex_t TVDevMutex
- #define TV_SERVICE_SERVCOUNT 2
- #define TV_SERVICE_CONTROL 0
- #define TV_SERVICE_PICTURE 1
- #define TV_CONTROL_VARCOUNT 3
- #define TV_CONTROL_POWER 0
- #define TV_CONTROL_CHANNEL 1
- #define TV_CONTROL_VOLUME 2
- #define TV_PICTURE_VARCOUNT 4
- #define TV_PICTURE_COLOR 0
- #define TV_PICTURE_TINT 1
- #define TV_PICTURE_CONTRAST 2
- #define TV_PICTURE_BRIGHTNESS 3
- #define TV_MAX_VAL_LEN 5
- #define TV_MAXVARS TV_PICTURE_VARCOUNT
- #define MAX_COLOR 10
- #define **MIN_COLOR** 1
- #define MAX_BRIGHTNESS 10
- #define **MIN_BRIGHTNESS** 1
- #define POWER_ON 1
- #define **POWER_OFF** 0
- #define MAX_TINT 10
- #define **MIN_TINT** 1
- #define MAX_VOLUME 10
- #define **MIN_VOLUME** 1
- #define MAX_CONTRAST 10
- #define **MIN_CONTRAST** 1
- #define MAX_CHANNEL 100
- #define **MIN_CHANNEL** 1
- #define MAX_COLOR 10
- #define **MIN_COLOR** 1
- #define MAX_BRIGHTNESS 10
- #define **MIN_BRIGHTNESS** 1
- #define POWER_ON 1
- #define **POWER_OFF** 0
- #define MAX_TINT 10
- #define **MIN_TINT** 1
- #define MAX_VOLUME 10
- #define **MIN_VOLUME** 1
- #define MAX_CONTRAST 10
- #define **MIN_CONTRAST** 1
- #define MAX_CHANNEL 100
- #define **MIN_CHANNEL** 1
- #define TV_SERVICE_SERVCOUNT 2
- #define TV_SERVICE_CONTROL 0
- #define TV_SERVICE_PICTURE 1
- #define TV_CONTROL_VARCOUNT 3
- #define TV_CONTROL_POWER 0

- #define TV_CONTROL_CHANNEL 1
- #define TV_CONTROL_VOLUME 2
- #define TV_PICTURE_VARCOUNT 4
- #define TV_PICTURE_COLOR 0
- #define TV_PICTURE_TINT 1
- #define TV_PICTURE_CONTRAST 2
- #define TV_PICTURE_BRIGHTNESS 3
- #define TV_MAX_VAL_LEN 5
- #define TV_MAXACTIONS 12
- #define TV_MAXVARS TV_PICTURE_VARCOUNT
- #define **IP_MODE_IPV4** 1
- #define **IP_MODE_IPV6_LLA** 2
- #define **IP_MODE_IPV6_ULA_GUA** 3

Contro Point Sample API

- const char * TvServiceName [ ]
- const char * TvVarName [2][4]
- char **TvVarCount** [ ]
- struct TvDeviceNode * GlobalDeviceList
- ithread_mutex_t DeviceListMutex
- UpnpClient_Handle **ctrlpt_handle**
- void **TvCtrlPointPrintHelp** (void)
- #define **TV_SUCCESS** 0
- #define **TV_ERROR** (-1)
- #define **TV_WARNING** 1

Device Sample Module

- const char * tvc_varname [ ] = {"Power", "Channel", "Volume"}
- char **tvc_varval** [3][5]
- const char * **tvc_varval_def** [ ] = {"1", "1", "5"}
- const char * tvp_varname [ ] = {"Color", "Tint", "Contrast", "Brightness"}
- char **tvp_varval** [4][5]
- const char * **tvp_varval_def** [ ] = {"5", "5", "5", "5"}
- int default_advr_expire = 100
- struct TvService tv_service_table [2]
- UpnpDevice_Handle device_handle = -1
- ithread_mutex_t TVDevMutex
- static int SetServiceTable (int serviceType, const char *UDN, const char *serviceId, const char *service↩ TypeS, struct TvService *out)

  Initializes the service table for the specified service.
- int SetActionTable (int serviceType, struct TvService *out)

  Initializes the action table for the specified service.
- int TvDeviceStateTableInit (char *DescDocURL)

  Initialize the device state table for this TvDevice, pulling identifier info from the description Document.
- int TvDeviceHandleSubscriptionRequest (const UpnpSubscriptionRequest *sr_event)

  Called during a subscription request callback.
- int TvDeviceHandleGetVarRequest (UpnpStateVarRequest *cgv_event)

  Called during a get variable request callback.
- int TvDeviceHandleActionRequest (UpnpActionRequest *ca_event)

  Called during an action request callback.

- int TvDeviceSetServiceTableVar (unsigned int service, int variable, char *value)

  Update the TvDevice service state table, and notify all subscribed control points of the updated state.
- static int TvDeviceSetPower (int on)

  Turn the power on/off, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- int TvDevicePowerOn (IXML_Document *in, IXML_Document **out, const char **errorString)

  Turn the power on.
- int TvDevicePowerOff (IXML_Document *in, IXML_Document **out, const char **errorString)

  Turn the power off.
- int TvDeviceSetChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the channel, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- int **IncrementChannel** (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)
- int TvDeviceDecreaseChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the channel.
- int TvDeviceIncreaseChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the channel.
- int TvDeviceSetVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the volume, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- static int IncrementVolume (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)

  Increment the volume. Read the current volume from the state table, add the increment, and then change the volume.
- int TvDeviceIncreaseVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the volume.
- int TvDeviceDecreaseVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the volume.
- int TvDeviceSetColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the color, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- static int IncrementColor (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)

  Increment the color. Read the current color from the state table, add the increment, and then change the color.
- int TvDeviceDecreaseColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the color.
- int TvDeviceIncreaseColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the color.
- int TvDeviceSetTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the tint, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- int **IncrementTint** (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)
- int TvDeviceIncreaseTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase tint.
- int TvDeviceDecreaseTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease tint.
- int TvDeviceSetContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the contrast, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- static int IncrementContrast (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)

  Increment the contrast. Read the current contrast from the state table, add the increment, and then change the contrast.
- int TvDeviceIncreaseContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the contrast.
- int TvDeviceDecreaseContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

Decrease the contrast.

- int TvDeviceSetBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the brightness, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- static int IncrementBrightness (int incr, IXML_Document *in, IXML_Document **out, const char **error↩ String)

  Increment the brightness. Read the current brightness from the state table, add the increment, and then change the brightness.
- int TvDeviceIncreaseBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase brightnesss.
- int TvDeviceDecreaseBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease brightnesss.
- int TvDeviceCallbackEventHandler (Upnp_EventType, const void *Event, void *Cookie)

  The callback handler registered with the SDK while registering root device.
- int TvDeviceStart (char *iface, unsigned short port, const char *desc_doc_name, const char *web_dir_path, int ip_mode, print_string pfun, int combo)

  Initializes the UPnP Sdk, registers the device, and sends out advertisements.
- int TvDeviceStop (void)

  Stops the device. Uninitializes the sdk.
- void * TvDeviceCommandLoop (void *args)

  Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands. Only one command, exit, is currently defined.
- int device_main (int argc, char *argv[ ])

  Main entry point for tv device application.
- #define **DEFAULT_WEB_DIR** "./web"
- #define **DESC_URL_SIZE** 200

7.4.1 Detailed Description

7.4.2 Macro Definition Documentation

7.4.2.1 MAX_BRIGHTNESS [1/2]

```
#define MAX_BRIGHTNESS 10
```

Brightness constants

7.4.2.2 MAX_BRIGHTNESS [2/2]

```
#define MAX_BRIGHTNESS 10
```

Brightness constants

7.4.2.3 MAX_CHANNEL [1/2]

```
#define MAX_CHANNEL 100
```

Channel constants

7.4.2.4 MAX_CHANNEL [2/2]

```
#define MAX_CHANNEL 100
```

Channel constants

7.4.2.5 MAX_COLOR [1/2]

```
#define MAX_COLOR 10
```

Color constants

7.4.2.6 MAX_COLOR [2/2]

```
#define MAX_COLOR 10
```

Color constants

7.4.2.7 MAX_CONTRAST [1/2]

```
#define MAX_CONTRAST 10
```

Contrast constants

7.4.2.8 MAX_CONTRAST [2/2]

```
#define MAX_CONTRAST 10
```

Contrast constants

7.4.2.9 MAX_TINT [1/2]

```
#define MAX_TINT 10
```

Tint constants

7.4.2.10 MAX_TINT [2/2]

```
#define MAX_TINT 10
```

Tint constants

7.4.2.11 MAX_VOLUME [1/2]

```
#define MAX_VOLUME 10
```

Volume constants

7.4.2.12 MAX_VOLUME [2/2]

```
#define MAX_VOLUME 10
```

Volume constants

7.4.2.13 POWER_ON [1/2]

```
#define POWER_ON 1
```

Power constants

7.4.2.14 POWER_ON [2/2]

```
#define POWER_ON 1
```

Power constants

7.4.2.15 TV_CONTROL_CHANNEL [1/2]

```
#define TV_CONTROL_CHANNEL 1
```

Index of channel variable

7.4.2.16 TV_CONTROL_CHANNEL [2/2]

```
#define TV_CONTROL_CHANNEL 1
```

Index of channel variable

7.4.2.17 TV_CONTROL_POWER [1/2]

```
#define TV_CONTROL_POWER 0
```

Index of power variable

7.4.2.18 TV_CONTROL_POWER [2/2]

```
#define TV_CONTROL_POWER 0
```

Index of power variable

7.4.2.19 TV_CONTROL_VARCOUNT [1/2]

```
#define TV_CONTROL_VARCOUNT 3
```

Number of control variables

7.4.2.20 TV_CONTROL_VARCOUNT [2/2]

```
#define TV_CONTROL_VARCOUNT 3
```

Number of control variables

7.4.2.21 TV_CONTROL_VOLUME [1/2]

```
#define TV_CONTROL_VOLUME 2
```

Index of volume variable

7.4.2.22 TV_CONTROL_VOLUME [2/2]

```
#define TV_CONTROL_VOLUME 2
```

Index of volume variable

7.4.2.23 TV_MAX_VAL_LEN [1/2]

```
#define TV_MAX_VAL_LEN 5
```

Max value length

7.4.2.24 TV_MAX_VAL_LEN [2/2]

```
#define TV_MAX_VAL_LEN 5
```

Max value length

7.4.2.25 TV_MAXACTIONS

```
#define TV_MAXACTIONS 12
```

Max actions

7.4.2.26 TV_MAXVARS [1/2]

```
#define TV_MAXVARS TV_PICTURE_VARCOUNT
```

This should be the maximum VARCOUNT from above

7.4.2.27 TV_MAXVARS [2/2]

```
#define TV_MAXVARS TV_PICTURE_VARCOUNT
```

This should be the maximum VARCOUNT from above

7.4.2.28 TV_PICTURE_BRIGHTNESS [1/2]

```
#define TV_PICTURE_BRIGHTNESS 3
```

Index of brightness variable

7.4.2.29 TV_PICTURE_BRIGHTNESS [2/2]

```
#define TV_PICTURE_BRIGHTNESS 3
```

Index of brightness variable

7.4.2.30 TV_PICTURE_COLOR [1/2]

```
#define TV_PICTURE_COLOR 0
```

Index of color variable

7.4.2.31 TV_PICTURE_COLOR [2/2]

```
#define TV_PICTURE_COLOR 0
```

Index of color variable

7.4.2.32 TV_PICTURE_CONTRAST [1/2]

```
#define TV_PICTURE_CONTRAST 2
```

Index of contrast variable

7.4.2.33 TV_PICTURE_CONTRAST [2/2]

```
#define TV_PICTURE_CONTRAST 2
```

Index of contrast variable

7.4.2.34 TV_PICTURE_TINT [1/2]

```
#define TV_PICTURE_TINT 1
```

Index of tint variable

7.4.2.35 TV_PICTURE_TINT [2/2]

```
#define TV_PICTURE_TINT 1
```

Index of tint variable

7.4.2.36 TV_PICTURE_VARCOUNT [1/2]

```
#define TV_PICTURE_VARCOUNT 4
```

Number of picture variables

7.4.2.37 TV_PICTURE_VARCOUNT [2/2]

```
#define TV_PICTURE_VARCOUNT 4
```

Number of picture variables

7.4.2.38 TV_SERVICE_CONTROL [1/2]

```
#define TV_SERVICE_CONTROL 0
```

Index of control service

7.4.2.39 TV_SERVICE_CONTROL [2/2]

```
#define TV_SERVICE_CONTROL 0
```

Index of control service

7.4.2.40 TV_SERVICE_PICTURE [1/2]

```
#define TV_SERVICE_PICTURE 1
```

Index of picture service

7.4.2.41 TV_SERVICE_PICTURE [2/2]

```
#define TV_SERVICE_PICTURE 1
```

Index of picture service

7.4.2.42 TV_SERVICE_SERVCOUNT [1/2]

```
#define TV_SERVICE_SERVCOUNT 2
```

Number of services.

7.4.2.43 TV_SERVICE_SERVCOUNT [2/2]

```
#define TV_SERVICE_SERVCOUNT 2
```

Number of services.

7.4.3 Typedef Documentation

7.4.3.1 print_string

```
typedef void(* print_string) ( const char *string, ...)
```

Prototype for displaying strings. All printing done by the device, control point, and sample util, ultimately use this to display strings to the user.

Parameters

in	*string*	Format.
in	...	Arguments.

7.4.3.2 state_update

```
typedef void(* state_update) ( const char *varName, const char *varValue, const char *UDN,
eventType type)
```

Prototype for passing back state changes.

Parameters

in	*varName*	.
in	*varValue*	.
in	*UDN*	.
in	*type*	.

7.4.3.3 upnp_action

```
typedef int(* upnp_action) ( IXML_Document *request, IXML_Document **out, const char **error↩
String)
```

Prototype for all actions. For each action that a service implements, there is a corresponding function with this prototype.

Pointers to these functions, along with action names, are stored in the service table. When an action request comes in the action name is matched, and the appropriate function is called. Each function returns UPNP_E_SUCCESS, on success, and a nonzero error code on failure.

Parameters

in	*request*	Document of action request.
out	*out*	Action result.
out	*errorString*	Error string in case action was unsuccessful.

7.4.4 Enumeration Type Documentation

7.4.4.1 cmdloop_tvcmds

```
enum cmdloop_tvcmds
```

Tags for valid commands issued at the command prompt.

7.4.5 Function Documentation

7.4.5.1 device_main()

```
int device_main (
            int argc,
            char * argv[] )
```

Main entry point for tv device application.

Initializes and registers with the sdk. Initializes the state stables of the service. Starts the command loop.

Accepts the following optional arguments:

- -ip ipaddress
- -port port
- -desc desc_doc_name
- -webdir web_dir_path
- -help

References linux_print(), SampleUtil_Initialize(), SampleUtil_Print(), and TvDeviceStart().

7.4.5.2 IncrementBrightness()

```
static int IncrementBrightness (
            int incr,
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )  [static]
```

Increment the brightness. Read the current brightness from the state table, add the increment, and then change the brightness.

Parameters

in	*incr*	The increment by which to change the brightness.
in	*in*	action request document.
out	*out*	action result document.
out	*errorString*	errorString (in case action was unsuccessful).

References MAX_BRIGHTNESS, SampleUtil_Print(), TV_MAX_VAL_LEN, TV_PICTURE_BRIGHTNESS, TV_SERVICE_PICTURE, tv_service_table, TvDeviceSetServiceTableVar(), TVDevMutex, UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UpnpAddToActionResponse(), and TvService::VariableStrVal.

Referenced by TvDeviceDecreaseBrightness(), and TvDeviceIncreaseBrightness().

7.4.5.3 IncrementColor()

```
static int IncrementColor (
            int incr,
```

```
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )  [static]
```

Increment the color. Read the current color from the state table, add the increment, and then change the color.

Parameters

in	*incr*	The increment by which to change the volume.
in	*in*	Action request document.
out	*out*	Action result document.
out	*errorString*	Error string in case action was unsuccessful.

References MAX_COLOR, SampleUtil_Print(), TV_MAX_VAL_LEN, TV_PICTURE_COLOR, TV_SERVICE_PICTURE, tv_service_table, TvDeviceSetServiceTableVar(), TVDevMutex, UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UpnpAddToActionResponse(), and TvService::VariableStrVal.

Referenced by TvDeviceDecreaseColor(), and TvDeviceIncreaseColor().

7.4.5.4 IncrementContrast()

```
static int IncrementContrast (
            int incr,
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )  [static]
```

Increment the contrast. Read the current contrast from the state table, add the increment, and then change the contrast.

Parameters

in	*incr*	The increment by which to change the volume.
in	*in*	Action request document.
out	*out*	Action result document.
out	*errorString*	Error string in case action was unsuccessful.

References MAX_CONTRAST, SampleUtil_Print(), TV_MAX_VAL_LEN, TV_PICTURE_CONTRAST, TV_SERVICE_PICTURE, tv_service_table, TvDeviceSetServiceTableVar(), TVDevMutex, UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UpnpAddToActionResponse(), and TvService::VariableStrVal.

Referenced by TvDeviceDecreaseContrast(), and TvDeviceIncreaseContrast().

7.4.5.5 IncrementVolume()

```
static int IncrementVolume (
            int incr,
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )  [static]
```

Increment the volume. Read the current volume from the state table, add the increment, and then change the volume.

Parameters

in	*incr*	The increment by which to change the volume.
in	*in*	Action request document.
out	*out*	Action result document.
out	*errorString*	Error string in case action was unsuccessful.

References MAX_VOLUME, SampleUtil_Print(), TV_CONTROL_VOLUME, TV_MAX_VAL_LEN, TV_SERVICE_CONTROL, tv_service_table, TvDeviceSetServiceTableVar(), TVDevMutex, UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UpnpAddToActionResponse(), and TvService::VariableStrVal.

Referenced by TvDeviceDecreaseVolume(), and TvDeviceIncreaseVolume().

7.4.5.6 linux_print()

```
void linux_print (
             const char * format,
              ...  )
```

Prints a string to standard out.

Referenced by device_main().

7.4.5.7 SampleUtil_FindAndParseService()

```
int SampleUtil_FindAndParseService (
             IXML_Document * DescDoc,
             const char * location,
             const char * serviceType,
             char ** serviceId,
             char ** eventURL,
             char ** controlURL )
```

This routine finds the first occurance of a service in a DOM representation of a description document and parses it. Note that this function currently assumes that the eventURL and controlURL values in the service definitions are full URLs. Relative URLs are not handled here.

Parameters

in	*DescDoc*	The DOM description document.
in	*location*	The location of the description document.
in	*serviceType*	The type of service to search for.
out	*serviceId*	The service ID.
out	*eventURL*	The event URL for the service.
out	*controlURL*	The control URL for the service.

References ixmlNodeList_free(), ixmlNodeList_item(), ixmlNodeList_length(), SampleUtil_GetFirstDocumentItem(), SampleUtil_GetFirstElementItem(), SampleUtil_GetFirstServiceList(), SampleUtil_Print(), UPNP_E_SUCCESS, and UpnpResolveURL2().

Referenced by TvDeviceStateTableInit().

7.4.5.8 SampleUtil_Finish()

```
int SampleUtil_Finish ( )
```

Releases Resources held by sample util.

References display_mutex, gPrintFun, gStateUpdateFun, and UPNP_E_SUCCESS.

Referenced by TvDeviceStop().

7.4.5.9 SampleUtil_GetElementValue()

```
char * SampleUtil_GetElementValue (
            IXML_Element * element )
```

Given a DOM node such as <Channel>11</Channel>, this routine extracts the value (e.g., 11) from the node and returns it as a string. The string must be freed by the caller using free.

Returns

The DOM node as a string.

Parameters

in	*element*	The DOM node from which to extract the value.

References ixmlNode_getFirstChild(), ixmlNode_getNodeType(), and ixmlNode_getNodeValue().

Referenced by TvStateUpdate().

7.4.5.10 SampleUtil_GetFirstDocumentItem()

```
char * SampleUtil_GetFirstDocumentItem (
            IXML_Document * doc,
            const char * item )
```

Given a document node, this routine searches for the first element named by the input string item, and returns its value as a string. String must be freed by caller using free.

Parameters

in	*doc*	The DOM document from which to extract the value.
in	*item*	The item to search for.

References ixmlDocument_getElementsByTagName(), ixmlNode_getFirstChild(), ixmlNode_getNodeValue(), ixmlNodeList_free(), ixmlNodeList_item(), and SampleUtil_Print().

Referenced by SampleUtil_FindAndParseService(), TvDeviceSetBrightness(), TvDeviceSetChannel(), TvDeviceSetColor(), TvDeviceSetContrast(), TvDeviceSetTint(), TvDeviceSetVolume(), and TvDeviceStateTableInit().

7.4.5.11 SampleUtil_GetFirstElementItem()

```
char * SampleUtil_GetFirstElementItem (
            IXML_Element * element,
            const char * item )
```

Given a DOM element, this routine searches for the first element named by the input string item, and returns its value as a string. The string must be freed using free.

Parameters

in	*element*	The DOM element from which to extract the value.
in	*item*	The item to search for.

References ixmlElement_getElementsByTagName(), ixmlNode_getFirstChild(), ixmlNode_getNodeValue(), ixmlNodeList_free(), ixmlNodeList_item(), and SampleUtil_Print().

Referenced by SampleUtil_FindAndParseService().

7.4.5.12 SampleUtil_GetFirstServiceList()

```
IXML_NodeList * SampleUtil_GetFirstServiceList (
            IXML_Document * doc )
```

Given a DOM node representing a UPnP Device Description Document, this routine parses the document and finds the first service list (i.e., the service list for the root device). The service list is returned as a DOM node list. The NodeList must be freed using NodeList_free.

Returns

The service list is returned as a DOM node list.

Parameters

in	*doc*	The DOM node from which to extract the service list.

References ixmlDocument_getElementsByTagName(), ixmlElement_getElementsByTagName(), ixmlNodeList_free(), ixmlNodeList_item(), and ixmlNodeList_length().

Referenced by SampleUtil_FindAndParseService().

7.4.5.13 SampleUtil_Initialize()

```
int SampleUtil_Initialize (
            print_string print_function )
```

Initializes the sample util. Must be called before any sample util functions. May be called multiple times.

Parameters

in	*print_function*	Print function to use in SampleUtil_Print.

References display_mutex, gPrintFun, and UPNP_E_SUCCESS.

Referenced by device_main(), and TvDeviceStart().

7.4.5.14 SampleUtil_Print()

```
int SampleUtil_Print (
             const char * fmt,
              ...  )
```

Function emulating printf that ultimately calls the registered print function with the formatted string.

Provides platform-specific print functionality. This function should be called when you want to print content suitable for console output (i.e., in a large text box or on a screen). If your device/operating system is not supported here, you should add a port.

Returns

The same as printf.

Parameters

in	*fmt*	Format (see printf).
in	...	Format data.

References display_mutex, and gPrintFun.

Referenced by device_main(), IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), SampleUtil_FindAndParseService(), SampleUtil_GetFirstDocumentItem(), SampleUtil_GetFirstElementItem(), SampleUtil_PrintEvent(), SampleUtil_PrintEventType(), TvCtrlPointCommandLoop(), TvCtrlPointPrintCommands(), TvCtrlPointPrintLongHelp(), TvCtrlPointPrintShortHelp(), TvCtrlPointStart(), TvCtrlPointVerifyTimeouts(), TvDeviceCallbackEventHand TvDeviceCommandLoop(), TvDeviceHandleGetVarRequest(), TvDeviceSetBrightness(), TvDeviceSetChannel(), TvDeviceSetColor(), TvDeviceSetContrast(), TvDeviceSetPower(), TvDeviceSetTint(), TvDeviceSetVolume(), TvDeviceStart(), TvDeviceStateTableInit(), and TvStateUpdate().

7.4.5.15 SampleUtil_PrintEvent()

```
int SampleUtil_PrintEvent (
             Upnp_EventType EventType,
             const void * Event )
```

Prints callback event structure details.

Parameters

in	*EventType*	The type of callback event.
in	*Event*	The callback event structure.

References display_mutex, ixmlFreeDOMString(), ixmlPrintNode(), SampleUtil_Print(), SampleUtil_PrintEventType(), UPNP_CONTROL_ACTION_COMPLETE, UPNP_CONTROL_ACTION_REQUEST, UPNP_CONTROL_GET_VAR_COMPLETE, UPNP_CONTROL_GET_VAR_REQUEST, UPNP_DISCOVERY_ADVERTISEMENT_ALIVE, UPNP_DISCOVERY_ADVERTISEMEN UPNP_DISCOVERY_SEARCH_RESULT, UPNP_DISCOVERY_SEARCH_TIMEOUT, UPNP_EVENT_AUTORENEWAL_FAILED, UPNP_EVENT_RECEIVED, UPNP_EVENT_RENEWAL_COMPLETE, UPNP_EVENT_SUBSCRIBE_COMPLETE, UPNP_EVENT_SUBSCRIPTION_EXPIRED, UPNP_EVENT_SUBSCRIPTION_REQUEST, UPNP_EVENT_UNSUBSCRIBE_COMP UpnpActionComplete_get_ActionRequest(), UpnpActionComplete_get_ActionResult(), UpnpActionComplete_get_CtrlUrl(), UpnpActionComplete_get_ErrCode(), UpnpActionRequest_get_ActionName(), UpnpActionRequest_get_ActionRequest(), UpnpActionRequest_get_ActionResult(), UpnpActionRequest_get_DevUDN(), UpnpActionRequest_get_ErrCode(), UpnpActionRequest_get_ErrStr(), UpnpActionRequest_get_ServiceID(), UpnpDiscovery_get_Date(), UpnpDiscovery_get_DeviceID() UpnpDiscovery_get_DeviceType(), UpnpDiscovery_get_ErrCode(), UpnpDiscovery_get_Expires(), UpnpDiscovery_get_Ext(), UpnpDiscovery_get_Location(), UpnpDiscovery_get_Os(), UpnpDiscovery_get_ServiceType(), UpnpDiscovery_get_ServiceVer(), UpnpEvent_get_ChangedVariables(), UpnpEvent_get_EventKey(), UpnpEvent_get_SID(), UpnpEventSubscribe_get_ErrCode(), UpnpEventSubscribe_get_PublisherUrl(), UpnpEventSubscribe_get_SID(), UpnpEventSubscribe_get_TimeOut(), UpnpStateVarComplete_get_CtrlUrl(), UpnpStateVarComplete_get_CurrentVal(), UpnpStateVarComplete_get_ErrCode(), UpnpStateVarComplete_get_StateVarName(), UpnpStateVarRequest_get_CurrentVal(), UpnpStateVarRequest_get_DevUDN(), UpnpStateVarRequest_get_ErrCode(), UpnpStateVarRequest_get_ErrStr(), UpnpStateVarRequest_get_ServiceID(), UpnpStateVarRequest_get_StateVarName(), UpnpString_get_String(), UpnpSubscriptionRequest_get_ServiceId(), UpnpSubscriptionRequest_get_SID(), and UpnpSubscriptionRequest_get_UDN().

Referenced by TvDeviceCallbackEventHandler().

7.4.5.16 SampleUtil_PrintEventType()

```
void SampleUtil_PrintEventType (
            Upnp_EventType S )
```

Prints a callback event type as a string.

Parameters

in	*S*	The callback event.

References SampleUtil_Print(), UPNP_CONTROL_ACTION_COMPLETE, UPNP_CONTROL_ACTION_REQUEST, UPNP_CONTROL_GET_VAR_COMPLETE, UPNP_CONTROL_GET_VAR_REQUEST, UPNP_DISCOVERY_ADVERTISEMENT_A UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE, UPNP_DISCOVERY_SEARCH_RESULT, UPNP_DISCOVERY_SEARCH_TIME UPNP_EVENT_AUTORENEWAL_FAILED, UPNP_EVENT_RECEIVED, UPNP_EVENT_RENEWAL_COMPLETE, UPNP_EVENT_SUBSCRIBE_COMPLETE, UPNP_EVENT_SUBSCRIPTION_EXPIRED, UPNP_EVENT_SUBSCRIPTION_REQUE and UPNP_EVENT_UNSUBSCRIBE_COMPLETE.

Referenced by SampleUtil_PrintEvent().

7.4.5.17 SampleUtil_RegisterUpdateFunction()

```
int SampleUtil_RegisterUpdateFunction (
            state_update update_function )
```

Parameters

in	*update_function*	.

References gStateUpdateFun, and UPNP_E_SUCCESS.

Referenced by TvCtrlPointStart().

7.4.5.18 SampleUtil_StateUpdate()

```
void SampleUtil_StateUpdate (
            const char * varName,
            const char * varValue,
            const char * UDN,
            eventType type )
```

Parameters

in	*varName*	.
in	*varValue*	.
in	*UDN*	.
in	*type*	.

References gStateUpdateFun.

7.4.5.19 SetActionTable()

```
int SetActionTable (
            int serviceType,
            struct TvService * out )
```

Initializes the action table for the specified service.

Note that knowledge of the service description is assumed. Action names are hardcoded.

Parameters

in	*serviceType*	one of TV_SERVICE_CONTROL or, TV_SERVICE_PICTURE.
in,out	*out*	service containing action table to set.

References TvService::ActionNames, TvService::actions, TV_SERVICE_CONTROL, TV_SERVICE_PICTURE, TvDeviceDecreaseBrightness(), TvDeviceDecreaseChannel(), TvDeviceDecreaseColor(), TvDeviceDecreaseContrast(), TvDeviceDecreaseTint(), TvDeviceDecreaseVolume(), TvDeviceIncreaseBrightness(), TvDeviceIncreaseChannel(), TvDeviceIncreaseColor(), TvDeviceIncreaseContrast(), TvDeviceIncreaseTint(), TvDeviceIncreaseVolume(), TvDevicePowerOff(), TvDevicePowerOn(), TvDeviceSetBrightness(), TvDeviceSetChannel(), TvDeviceSetColor(), TvDeviceSetContrast(), TvDeviceSetTint(), and TvDeviceSetVolume().

Referenced by SetServiceTable().

7.4.5.20 SetServiceTable()

```
static int SetServiceTable (
            int serviceType,
            const char * UDN,
            const char * serviceId,
```

```
            const char * serviceTypeS,
            struct TvService * out )  [static]
```

Initializes the service table for the specified service.

Parameters

in	*serviceType*	one of TV_SERVICE_CONTROL or, TV_SERVICE_PICTURE.
in	*UDN*	UDN of device containing service.
in	*serviceId*	serviceId of service.
in	*serviceTypeS*	service type (as specified in Description Document) .
in,out	*out*	service containing table to be set.

References TvService::ServiceId, TvService::ServiceType, SetActionTable(), TV_CONTROL_VARCOUNT, TV_PICTURE_VARCOUNT, TV_SERVICE_CONTROL, TV_SERVICE_PICTURE, tv_service_table, tvc_varname, tvp_varname, TvService::UDN, TvService::VariableCount, TvService::VariableName, and TvService::VariableStrVal.

Referenced by TvDeviceStateTableInit().

7.4.5.21 TvCtrlPointCommandLoop()

```
void * TvCtrlPointCommandLoop (
            void * args )
```

Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands.

References SampleUtil_Print().

7.4.5.22 TvCtrlPointPrintCommands()

```
void TvCtrlPointPrintCommands (
            void  )
```

\briefPrint the list of valid command line commands to the user

References cmdloop_cmdlist, and SampleUtil_Print().

7.4.5.23 TvCtrlPointPrintLongHelp()

```
void TvCtrlPointPrintLongHelp (
            void  )
```

Print long help info for this application.

References SampleUtil_Print().

7.4.5.24 TvCtrlPointPrintShortHelp()

```
void TvCtrlPointPrintShortHelp (
            void  )
```

Print help info for this application.

References SampleUtil_Print().

7.4.5.25 TvCtrlPointStart()

```
int TvCtrlPointStart (
              char * iface,
              state_update updateFunctionPtr,
              int combo )
```

Call this function to initialize the UPnP library and start the TV Control Point. This function creates a timer thread and provides a callback handler to process any UPnP events that are received.

Returns

TV_SUCCESS if everything went well, else TV_ERROR.

References DeviceListMutex, SampleUtil_Print(), SampleUtil_RegisterUpdateFunction(), UPNP_E_SUCCESS, UpnpFinish(), UpnpGetServerIp6Address(), UpnpGetServerIpAddress(), UpnpGetServerPort(), UpnpGetServerPort6(), UpnpGetServerUlaGuaIp6Address(), UpnpGetServerUlaGuaPort6(), UpnpInit2(), and UpnpRegisterClient().

7.4.5.26 TvCtrlPointVerifyTimeouts()

```
void TvCtrlPointVerifyTimeouts (
              int incr )
```

Checks the advertisement each device in the global device list.

If an advertisement expires, the device is removed from the list.

If an advertisement is about to expire, a search request is sent for that device.

Parameters

in	*incr*	The increment to subtract from the timeouts each time the function is called.

References DeviceListMutex, GlobalDeviceList, SampleUtil_Print(), UPNP_E_SUCCESS, and UpnpSearchAsync().

7.4.5.27 TvDeviceCallbackEventHandler()

```
int TvDeviceCallbackEventHandler (
              Upnp_EventType ,
              const void * Event,
              void * Cookie )
```

The callback handler registered with the SDK while registering root device.

Dispatches the request to the appropriate procedure based on the value of EventType. The four requests handled by the device are:

- 1) Event Subscription requests.
- 2) Get Variable requests.
- 3) Action requests.

Parameters

in	*EventType*	The type of callback event.
in	*Event*	Data structure containing event data.
in	*Cookie*	Optional data specified during callback registration.

References SampleUtil_Print(), SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), UPNP_CONTROL_ACTION_COMPLETE, UPNP_CONTROL_ACTION_REQUEST, UPNP_CONTROL_GET_VAR_COMPLETE, UPNP_CONTROL_GET_VAR_REQUEST, UPNP_DISCOVERY_ADVERTISEMENT_A UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE, UPNP_DISCOVERY_SEARCH_RESULT, UPNP_DISCOVERY_SEARCH_TIME UPNP_EVENT_RECEIVED, UPNP_EVENT_RENEWAL_COMPLETE, UPNP_EVENT_SUBSCRIBE_COMPLETE, UPNP_EVENT_SUBSCRIPTION_REQUEST, and UPNP_EVENT_UNSUBSCRIBE_COMPLETE.

Referenced by TvDeviceStart().

7.4.5.28 TvDeviceCommandLoop()

```
void * TvDeviceCommandLoop (
            void * args )
```

Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands. Only one command, exit, is currently defined.

References SampleUtil_Print(), and TvDeviceStop().

7.4.5.29 TvDeviceDecreaseBrightness()

```
int TvDeviceDecreaseBrightness (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Decrease brightnesss.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementBrightness().

Referenced by SetActionTable().

7.4.5.30 TvDeviceDecreaseChannel()

```
int TvDeviceDecreaseChannel (
            IXML_Document * in,
```

```
            IXML_Document ** out,
            const char ** errorString )
```

Decrease the channel.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

Referenced by SetActionTable().

7.4.5.31 TvDeviceDecreaseColor()

```
int TvDeviceDecreaseColor (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Decrease the color.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementColor().

Referenced by SetActionTable().

7.4.5.32 TvDeviceDecreaseContrast()

```
int TvDeviceDecreaseContrast (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Decrease the contrast.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementContrast().

Referenced by SetActionTable().

7.4.5.33 TvDeviceDecreaseTint()

```
int TvDeviceDecreaseTint (
            IXML_Document * in,
```

```
            IXML_Document ** out,
            const char ** errorString )
```

Decrease tint.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

Referenced by SetActionTable().

7.4.5.34 TvDeviceDecreaseVolume()

```
int TvDeviceDecreaseVolume (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Decrease the volume.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementVolume().

Referenced by SetActionTable().

7.4.5.35 TvDeviceHandleActionRequest()

```
int TvDeviceHandleActionRequest (
            UpnpActionRequest * ca_event )
```

Called during an action request callback.

If the request is for this device and either its control service or picture service, then perform the action and respond.

Parameters

in,out	*ca_event*	The control action request event structure.

References TvService::ActionNames, TvService::actions, TvService::ServiceId, TV_CONTROL_POWER, TV_MAXACTIONS, TV_SERVICE_CONTROL, TV_SERVICE_PICTURE, tv_service_table, TvService::UDN, UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UpnpActionRequest_get_ActionName(), UpnpActionRequest_get_ActionRequest(), UpnpActionRequest_get_DevUDN(), UpnpActionRequest_get_ErrCode(),

UpnpActionRequest_get_ServiceID(), UpnpActionRequest_set_ActionResult(), UpnpActionRequest_set_ErrCode(), UpnpActionRequest_strcpy_ErrStr(), UpnpString_get_String(), and TvService::VariableStrVal.

Referenced by TvDeviceCallbackEventHandler().

7.4.5.36 TvDeviceHandleGetVarRequest()

```
int TvDeviceHandleGetVarRequest (
            UpnpStateVarRequest * cgv_event )
```

Called during a get variable request callback.

If the request is for this device and either its control service or picture service, then respond with the variable value.

Parameters

in,out	*cgv_event*	The control get variable request event structure.

References SampleUtil_Print(), TvService::ServiceId, TV_SERVICE_SERVCOUNT, tv_service_table, TVDevMutex, TvService::UDN, UPNP_E_SUCCESS, UpnpStateVarRequest_get_DevUDN(), UpnpStateVarRequest_get_ErrCode(), UpnpStateVarRequest_get_ServiceID(), UpnpStateVarRequest_get_StateVarName(), UpnpStateVarRequest_set_CurrentVal(), UpnpStateVarRequest_set_ErrCode(), UpnpStateVarRequest_strcpy_ErrStr(), UpnpString_get_String(), TvService::VariableCount, TvService::VariableName, and TvService::VariableStrVal.

Referenced by TvDeviceCallbackEventHandler().

7.4.5.37 TvDeviceHandleSubscriptionRequest()

```
int TvDeviceHandleSubscriptionRequest (
            const UpnpSubscriptionRequest * sr_event )
```

Called during a subscription request callback.

If the subscription request is for this device and either its control service or picture service, then accept it.

Parameters

in	*sr_event*	The subscription request event structure.

References device_handle, TvService::ServiceId, TV_SERVICE_SERVCOUNT, tv_service_table, TVDevMutex, TvService::UDN, UpnpAcceptSubscription(), UpnpAcceptSubscriptionExt(), UpnpAddToPropertySet(), UpnpString_get_String(), UpnpSubscriptionRequest_get_ServiceId(), UpnpSubscriptionRequest_get_SID_cstr(), UpnpSubscriptionRequest_get_UDN_cstr(), TvService::VariableCount, TvService::VariableName, and TvService::VariableStrVal.

Referenced by TvDeviceCallbackEventHandler().

7.4.5.38 TvDeviceIncreaseBrightness()

```
int TvDeviceIncreaseBrightness (
            IXML_Document * in,
```

```
            IXML_Document ** out,
            const char ** errorString )
```

Increase brightnesss.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementBrightness().

Referenced by SetActionTable().

7.4.5.39 TvDeviceIncreaseChannel()

```
int TvDeviceIncreaseChannel (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Increase the channel.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

Referenced by SetActionTable().

7.4.5.40 TvDeviceIncreaseColor()

```
int TvDeviceIncreaseColor (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Increase the color.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementColor().

Referenced by SetActionTable().

7.4.5.41 TvDeviceIncreaseContrast()

```
int TvDeviceIncreaseContrast (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Increase the contrast.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementContrast().

Referenced by SetActionTable().

7.4.5.42 TvDeviceIncreaseTint()

```
int TvDeviceIncreaseTint (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Increase tint.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

Referenced by SetActionTable().

7.4.5.43 TvDeviceIncreaseVolume()

```
int TvDeviceIncreaseVolume (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Increase the volume.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References IncrementVolume().

Referenced by SetActionTable().

7.4.5.44 TvDevicePowerOff()

```
int TvDevicePowerOff (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Turn the power off.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References TV_SERVICE_CONTROL, TvDeviceSetPower(), UPNP_E_INTERNAL_ERROR, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.45 TvDevicePowerOn()

```
int TvDevicePowerOn (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Turn the power on.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References POWER_ON, TV_SERVICE_CONTROL, TvDeviceSetPower(), UPNP_E_INTERNAL_ERROR, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.46 TvDeviceSetBrightness()

```
int TvDeviceSetBrightness (
            IXML_Document * in,
```

```
            IXML_Document ** out,
            const char ** errorString )
```

Change the brightness, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References MAX_BRIGHTNESS, SampleUtil_GetFirstDocumentItem(), SampleUtil_Print(), TV_PICTURE_BRIGHTNESS, TV_SERVICE_PICTURE, TvDeviceSetServiceTableVar(), UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.47 TvDeviceSetChannel()

```
int TvDeviceSetChannel (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Change the channel, update the TvDevice control service state table, and notify all subscribed control points of the updated state.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References MAX_CHANNEL, SampleUtil_GetFirstDocumentItem(), SampleUtil_Print(), TV_CONTROL_CHANNEL, TV_SERVICE_CONTROL, TvDeviceSetServiceTableVar(), UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.48 TvDeviceSetColor()

```
int TvDeviceSetColor (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Change the color, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References MAX_COLOR, SampleUtil_GetFirstDocumentItem(), SampleUtil_Print(), TV_PICTURE_COLOR, TV_SERVICE_PICTURE, TvDeviceSetServiceTableVar(), UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.49 TvDeviceSetContrast()

```
int TvDeviceSetContrast (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Change the contrast, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References MAX_CONTRAST, SampleUtil_GetFirstDocumentItem(), SampleUtil_Print(), TV_PICTURE_CONTRAST, TV_SERVICE_PICTURE, TvDeviceSetServiceTableVar(), UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.50 TvDeviceSetPower()

```
static int TvDeviceSetPower (
            int on )  [static]
```

Turn the power on/off, update the TvDevice control service state table, and notify all subscribed control points of the updated state.

Parameters

in	*on*	If 1, turn power on. If 0, turn power off.

References POWER_ON, SampleUtil_Print(), TV_CONTROL_POWER, TV_MAX_VAL_LEN, TV_SERVICE_CONTROL, and TvDeviceSetServiceTableVar().

Referenced by TvDevicePowerOff(), and TvDevicePowerOn().

7.4.5.51 TvDeviceSetServiceTableVar()

```
int TvDeviceSetServiceTableVar (
            unsigned int service,
            int variable,
            char * value )
```

Update the TvDevice service state table, and notify all subscribed control points of the updated state.

Note that since this function blocks on the mutex TVDevMutex, to avoid a hang this function should not be called within any other function that currently has this mutex locked.

Parameters

in	*service*	The service number (TV_SERVICE_CONTROL or TV_SERVICE_PICTURE).
in	*variable*	The variable number (TV_CONTROL_POWER, TV_CONTROL_CHANNEL, TV_CONTROL_VOLUME, TV_PICTURE_COLOR, TV_PICTURE_TINT, TV_PICTURE_CONTRAST, or TV_PICTURE_BRIGHTNESS).
in	*value*	The string representation of the new value.

References device_handle, TvService::ServiceId, TV_MAX_VAL_LEN, TV_SERVICE_SERVCOUNT, tv_service_table, TVDevMutex, TvService::UDN, UpnpCreatePropertySet(), UpnpNotify(), UpnpNotifyExt(), TvService::VariableCount, TvService::VariableName, and TvService::VariableStrVal.

Referenced by IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), TvDeviceSetBrightness(), TvDeviceSetChannel(), TvDeviceSetColor(), TvDeviceSetContrast(), TvDeviceSetPower(), TvDeviceSetTint(), and TvDeviceSetVolume().

7.4.5.52 TvDeviceSetTint()

```
int TvDeviceSetTint (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Change the tint, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References MAX_TINT, SampleUtil_GetFirstDocumentItem(), SampleUtil_Print(), TV_PICTURE_TINT, TV_SERVICE_PICTURE, TvDeviceSetServiceTableVar(), UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.53 TvDeviceSetVolume()

```
int TvDeviceSetVolume (
            IXML_Document * in,
            IXML_Document ** out,
            const char ** errorString )
```

Change the volume, update the TvDevice control service state table, and notify all subscribed control points of the updated state.

Parameters

in	*in*	Document of action request.
in	*out*	Action result.
out	*errorString*	ErrorString in case action was unsuccessful.

References MAX_VOLUME, SampleUtil_GetFirstDocumentItem(), SampleUtil_Print(), TV_CONTROL_VOLUME, TV_SERVICE_CONTROL, TvDeviceSetServiceTableVar(), UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpAddToActionResponse().

Referenced by SetActionTable().

7.4.5.54 TvDeviceStart()

```
int TvDeviceStart (
            char * iface,
            unsigned short port,
            const char * desc_doc_name,
            const char * web_dir_path,
            int ip_mode,
            print_string pfun,
            int combo )
```

Initializes the UPnP Sdk, registers the device, and sends out advertisements.

Parameters

in	*iface*	interface to initialize the sdk (may be NULL) if null, then the first non null interface is used.
in	*port*	port number to initialize the sdk (may be 0) if zero, then a random number is used.
in	*desc_doc_name*	name of description document. may be NULL. Default is tvdevicedesc.xml.
in	*web_dir_path*	path of web directory. may be NULL. Default is ./web (for Linux) or ../tvdevice/web.
in	*ip_mode*	IP mode: IP_MODE_IPV4, IP_MODE_IPV6_LLA or IP_MODE_IPV6_ULA_GUA. Default is IP_MODE_IPV4.
in	*pfun*	print function to use.
in	*combo*	Non-zero if called from the combo application.

References default_advr_expire, device_handle, SampleUtil_Initialize(), SampleUtil_Print(), TvDeviceCallbackEventHandler(), TvDeviceStateTableInit(), TVDevMutex, UPNP_E_INTERNAL_ERROR, UPNP_E_SUCCESS, UpnpFinish(), UpnpGetServerIp6Address(), UpnpGetServerIpAddress(), UpnpGetServerPort(), UpnpGetServerPort6(), UpnpGetServerUlaGuaIp6Address(), UpnpGetServerUlaGuaPort6(), UpnpInit2(), UpnpInitLog(), UpnpRegisterRootDevice3(), UpnpSendAdvertisement(), UpnpSetLogFileNames(), UpnpSetLogLevel(), and UpnpSetWebServerRootDir().

Referenced by device_main().

7.4.5.55 TvDeviceStateTableInit()

```
int TvDeviceStateTableInit (
            char * DescDocURL )
```

Initialize the device state table for this TvDevice, pulling identifier info from the description Document.

Note that knowledge of the service description is assumed. State table variables and default values are currently hardcoded in this file rather than being read from service description documents.

Parameters

in	*DescDocURL*	The description document URL.

References ixmlDocument_free(), SampleUtil_FindAndParseService(), SampleUtil_GetFirstDocumentItem(), SampleUtil_Print(), SetServiceTable(), TV_SERVICE_CONTROL, TV_SERVICE_PICTURE, tv_service_table, UPNP_E_INVALID_DESC, UPNP_E_SUCCESS, and UpnpDownloadXmlDoc().

Referenced by TvDeviceStart().

7.4.5.56 TvDeviceStop()

```
int TvDeviceStop (
            void  )
```

Stops the device. Uninitializes the sdk.

References device_handle, SampleUtil_Finish(), TVDevMutex, UPNP_E_SUCCESS, UpnpFinish(), and UpnpUnRegisterRootDevice().

Referenced by TvDeviceCommandLoop().

7.4.5.57 TvStateUpdate()

```
void TvStateUpdate (
            char * UDN,
            int Service,
            IXML_Document * ChangedVariables,
            char ** State )
```

Update a Tv state table. Called when an event is received.

Note: this function is NOT thread save. It must be called from another function that has locked the global device list.

Parameters

in	*UDN*	The UDN of the parent device.
in	*Service*	The service state table to update.
out	*ChangedVariables*	DOM document representing the XML received with the event.
out	*State*	pointer to the state table for the Tv service to update.

References ixmlDocument_getElementsByTagName(), ixmlElement_getElementsByTagName(), ixmlNodeList_free(), ixmlNodeList_item(), ixmlNodeList_length(), SampleUtil_GetElementValue(), SampleUtil_Print(), and TvVarName.

7.4.6 Variable Documentation

7.4.6.1 cmdloop_cmdlist

```
struct cmdloop_commands cmdloop_cmdlist[]  [static]
```

Initial value:

```
= {{"Help", PRTHELP, 1, ""},
        {"HelpFull", PRTFULLHELP, 1, ""},
        {"ListDev", LSTDEV, 1, ""},
        {"Refresh", REFRESH, 1, ""},
        {"PrintDev", PRTDEV, 2, "<devnum>"},
        {"PowerOn", POWON, 2, "<devnum>"},
        {"PowerOff", POWOFF, 2, "<devnum>"},
        {"SetChannel", SETCHAN, 3, "<devnum> <channel (int)>"},
        {"SetVolume", SETVOL, 3, "<devnum> <volume (int)>"},
        {"SetColor", SETCOL, 3, "<devnum> <color (int)>"},
        {"SetTint", SETTINT, 3, "<devnum> <tint (int)>"},
        {"SetContrast", SETCONT, 3, "<devnum> <contrast (int)>"},
        {"SetBrightness", SETBRT, 3, "<devnum> <brightness (int)>"},
        {"CtrlAction", CTRLACTION, 2, "<devnum> <action (string)>"},
        {"PictAction", PICTACTION, 2, "<devnum> <action (string)>"},
        {"CtrlGetVar", CTRLGETVAR, 2, "<devnum> <varname (string)>"},
        {"PictGetVar", PICTGETVAR, 2, "<devnum> <varname (string)>"},
        {"Exit", EXITCMD, 1, ""}}
```

Mappings between command text names, command tag, and required command arguments for command line commands

Referenced by TvCtrlPointPrintCommands().

7.4.6.2 default_advr_expire

```
int default_advr_expire = 100
```

The amount of time (in seconds) before advertisements will expire.

Referenced by TvDeviceStart().

7.4.6.3 default_timeout

```
int default_timeout = 1801
```

Timeout to request during subscriptions

7.4.6.4 device_handle [1/2]

```
UpnpDevice_Handle device_handle = -1
```

Device handle supplied by UPnP SDK.

Referenced by gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_re genaInitNotify(), genaInitNotifyExt(), genaNotifyAll(), genaNotifyAllExt(), genaUnregisterDevice(), TvDeviceHandleSubscriptionReques TvDeviceSetServiceTableVar(), TvDeviceStart(), TvDeviceStop(), and UpnpFinish().

7.4.6.5 device_handle [2/2]

```
UpnpDevice_Handle device_handle  [extern]
```

Device handle returned from sdk

Device handle supplied by UPnP SDK.

Referenced by gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_re genaInitNotify(), genaInitNotifyExt(), genaNotifyAll(), genaNotifyAllExt(), genaUnregisterDevice(), TvDeviceHandleSubscriptionReques TvDeviceSetServiceTableVar(), TvDeviceStart(), TvDeviceStop(), and UpnpFinish().

7.4.6.6 DeviceListMutex [1/2]

```
ithread_mutex_t DeviceListMutex
```

Mutex for protecting the global device list in a multi-threaded, asynchronous environment. All functions should lock this mutex before reading or writing the device list.

Referenced by TvCtrlPointStart(), and TvCtrlPointVerifyTimeouts().

7.4.6.7 DeviceListMutex [2/2]

```
ithread_mutex_t DeviceListMutex  [extern]
```

Mutex for protecting the global device list in a multi-threaded, asynchronous environment. All functions should lock this mutex before reading or writing the device list.

Referenced by TvCtrlPointStart(), and TvCtrlPointVerifyTimeouts().

7.4.6.8 display_mutex [1/2]

```
ithread_mutex_t display_mutex
```

mutex to control displaying of events

Referenced by SampleUtil_Finish(), SampleUtil_Initialize(), SampleUtil_Print(), and SampleUtil_PrintEvent().

7.4.6.9 display_mutex [2/2]

```
ithread_mutex_t display_mutex  [extern]
```

mutex to control displaying of events

Referenced by SampleUtil_Finish(), SampleUtil_Initialize(), SampleUtil_Print(), and SampleUtil_PrintEvent().

7.4.6.10 GlobalDeviceList [1/2]

```
struct TvDeviceNode* GlobalDeviceList = NULL
```

The first node in the global device list, or NULL if empty

Referenced by TvCtrlPointVerifyTimeouts().

7.4.6.11 GlobalDeviceList [2/2]

```
struct TvDeviceNode* GlobalDeviceList  [extern]
```

The first node in the global device list, or NULL if empty

Referenced by TvCtrlPointVerifyTimeouts().

7.4.6.12 gPrintFun [1/2]

```
print_string gPrintFun = NULL
```

Function pointers to use for displaying formatted strings. Set on Initialization of device.

Referenced by SampleUtil_Finish(), SampleUtil_Initialize(), and SampleUtil_Print().

7.4.6.13 gPrintFun [2/2]

```
print_string gPrintFun  [extern]
```

global print function used by sample util

Function pointers to use for displaying formatted strings. Set on Initialization of device.

Referenced by SampleUtil_Finish(), SampleUtil_Initialize(), and SampleUtil_Print().

7.4.6.14 gStateUpdateFun [1/2]

```
state_update gStateUpdateFun = NULL
```

global state update function used by smaple util

Referenced by SampleUtil_Finish(), SampleUtil_RegisterUpdateFunction(), and SampleUtil_StateUpdate().

7.4.6.15 gStateUpdateFun [2/2]

```
state_update gStateUpdateFun  [extern]
```

global state update function used by smaple util

Referenced by SampleUtil_Finish(), SampleUtil_RegisterUpdateFunction(), and SampleUtil_StateUpdate().

7.4.6.16 tv_service_table [1/2]

```
struct TvService tv_service_table[2]
```

Global structure for storing the state table for this device.

Referenced by IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), SetServiceTable(), TvDeviceHandleActionRequest(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), TvDeviceSetServiceTableVar(), and TvDeviceStateTableInit().

7.4.6.17 tv_service_table [2/2]

```
struct TvService tv_service_table[]  [extern]
```

Array of service structures

Global structure for storing the state table for this device.

Referenced by IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), SetServiceTable(), TvDeviceHandleActionRequest(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), TvDeviceSetServiceTableVar(), and TvDeviceStateTableInit().

7.4.6.18 tvc_varname

```
const char* tvc_varname[] = {"Power", "Channel", "Volume"}
```

Global arrays for storing Tv Control Service variable names, values, and defaults.

Referenced by SetServiceTable().

7.4.6.19 TvCtrlPointTimerLoopRun

```
int TvCtrlPointTimerLoopRun = 1  [static]
```

Function that runs in its own thread and monitors advertisement and subscription timeouts for devices in the global device list.

7.4.6.20 TvDeviceType

```
const char TvDeviceType[] = "urn:schemas-upnp-org:device:tvdevice:1"
```

Device type for tv device.

7.4.6.21 TVDevMutex [1/2]

```
ithread_mutex_t TVDevMutex
```

Mutex for protecting the global state table data in a multi-threaded, asynchronous environment. All functions should lock this mutex before reading or writing the state table data.

Referenced by IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), TvDeviceSetServiceTableVar(), TvDeviceStart(), and TvDeviceStop().

7.4.6.22 TVDevMutex [2/2]

```
ithread_mutex_t TVDevMutex  [extern]
```

Mutex for protecting the global state table data in a multi-threaded, asynchronous environment. All functions should lock this mutex before reading or writing the state table data.

Referenced by IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), TvDeviceSetServiceTableVar(), TvDeviceStart(), and TvDeviceStop().

7.4.6.23 tvp_varname

```
const char* tvp_varname[] = {"Color", "Tint", "Contrast", "Brightness"}
```

Global arrays for storing Tv Picture Service variable names, values, and defaults.

Referenced by SetServiceTable().

7.4.6.24 TvServiceName [1/2]

```
const char* TvServiceName[] = {"Control", "Picture"}
```

Service names.

7.4.6.25 TvServiceName [2/2]

```
const char* TvServiceName[]  [extern]
```

Service names.

7.4.6.26 TvVarCount

```
char TvVarCount[2]
```

Initial value:

```
= {
        3 ,  4 }
```

7.4.6.27 TvVarName [1/2]

```
const char* TvVarName[2][ 4]
```

Initial value:

```
= {
       {"Power", "Channel", "Volume", ""},
       {"Color", "Tint", "Contrast", "Brightness"}}
```

Global arrays for storing variable names and counts for TvControl and TvPicture services

Referenced by TvStateUpdate().

7.4.6.28 TvVarName [2/2]

```
const char* TvVarName[2][ 4]  [extern]
```

Global arrays for storing variable names and counts for TvControl and TvPicture services

Referenced by TvStateUpdate().

7.5 Network Socket Library

Files

- file UpnpInet.h

  Provides a platform independent way to include TCP/IP types and functions.
- file sock.c

  Implements the sockets functionality.
- file sock.h

Data Structures

- struct SOCKINFO

Macros

- #define INVALID_SOCKET (-1)
- #define SOCKET_ERROR (-1)
- #define UpnpCloseSocket close
- #define **MSG_NOSIGNAL** 0
- #define **SD_RECEIVE** 0x00
- #define **SD_SEND** 0x01
- #define **SD_BOTH** 0x02

Typedefs

- typedef int SOCKET

Functions

- int sock_init (SOCKINFO *info, SOCKET sockfd)

  Assign the passed in socket descriptor to socket descriptor in the SOCKINFO structure.
- int sock_init_with_ip (SOCKINFO *info, SOCKET sockfd, struct sockaddr *foreign_sockaddr)

  Calls the sock_init function and assigns the passed in IP address and port to the IP address and port in the SOCKINFO structure.
- int sock_destroy (SOCKINFO *info, int ShutdownMethod)

  Associates an SSL object with the socket and begins the client-side SSL/TLS handshake.
- static int sock_read_write (SOCKINFO *info, char *buffer, size_t bufsize, int *timeoutSecs, int bRead)

  Receives or sends data. Also returns the time taken to receive or send data.
- int sock_read (SOCKINFO *info, char *buffer, size_t bufsize, int *timeoutSecs)

  Reads data on socket in sockinfo.
- int sock_write (SOCKINFO *info, const char *buffer, size_t bufsize, int *timeoutSecs)

  Writes data on the socket in sockinfo.
- int sock_make_blocking (SOCKET sock)

  Make socket blocking.
- int sock_make_no_blocking (SOCKET sock)

  Make socket non-blocking.
- static UPNP_INLINE int sock_close (SOCKET sock)

  Closes the socket if it is different from -1.

Variables

- SOCKET SOCKINFO::socket
- struct sockaddr_storage SOCKINFO::foreign_sockaddr

7.5.1 Detailed Description

7.5.2 Macro Definition Documentation

7.5.2.1 INVALID_SOCKET

```
#define INVALID_SOCKET (-1)
```

INVALID_SOCKET is unsigned on win32.

7.5.2.2 SOCKET_ERROR

```
#define SOCKET_ERROR (-1)
```

select() returns SOCKET_ERROR on win32.

7.5.2.3 UpnpCloseSocket

```
#define UpnpCloseSocket close
```

Alias to close() to make code more WIN32 tolerant.

7.5.3 Typedef Documentation

7.5.3.1 SOCKET

```
typedef int SOCKET
```

This typedef makes the code slightly more WIN32 tolerant. On WIN32 systems, SOCKET is unsigned and is not a file descriptor.

7.5.4 Function Documentation

7.5.4.1 sock_close()

```
static UPNP_INLINE int sock_close (
            SOCKET sock )  [static]
```

Closes the socket if it is different from -1.

Returns

-1 if an error occurred or if the socket is -1.

Parameters

sock	Socket descriptor.

References INVALID_SOCKET, and UpnpCloseSocket.

Referenced by get_miniserver_stopsock(), RunMiniServer(), sock_destroy(), StartMiniServer(), and StopMiniServer().

7.5.4.2 sock_destroy()

```
int sock_destroy (
            SOCKINFO * info,
            int ShutdownMethod )
```

Associates an SSL object with the socket and begins the client-side SSL/TLS handshake.

Returns

Integer:

- UPNP_E_SUCCESS
- UPNP_E_SOCKET_ERROR

Shutsdown the socket using the ShutdownMethod to indicate whether sends and receives on the socket will be dis-allowed.

After shutting down the socket, closesocket is called to release system resources used by the socket calls.

Returns

Integer:

- UPNP_E_SOCKET_ERROR on failure.
- UPNP_E_SUCCESS on success.

Parameters

in, out	*info*	Socket Information Object.
in	*ShutdownMethod*	How to shutdown the socket. Used by sockets's shutdown().

References INVALID_SOCKET, sock_close(), SOCKINFO::socket, UPNP_E_SOCKET_ERROR, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by http_CloseHttpConnection(), http_OpenHttpConnection(), and notify_send_and_recv().

7.5.4.3 sock_init()

```
int sock_init (
            SOCKINFO * info,
            SOCKET sockfd )
```

Assign the passed in socket descriptor to socket descriptor in the SOCKINFO structure.

Returns

Integer:

- UPNP_E_SUCCESS
- UPNP_E_OUTOF_MEMORY
- UPNP_E_SOCKET_ERROR

Parameters

out	*info*	Socket Information Object.
in	*sockfd*	Socket Descriptor.

References SOCKINFO::socket, and UPNP_E_SUCCESS.

Referenced by http_OpenHttpConnection(), notify_send_and_recv(), and sock_init_with_ip().

7.5.4.4 sock_init_with_ip()

```
int sock_init_with_ip (
            SOCKINFO * info,
            SOCKET sockfd,
            struct sockaddr * foreign_sockaddr )
```

Calls the sock_init function and assigns the passed in IP address and port to the IP address and port in the SOCKINFO structure.

Returns

Integer:

- UPNP_E_SUCCESS
- UPNP_E_OUTOF_MEMORY
- UPNP_E_SOCKET_ERROR

Parameters

out	*info*	Socket Information Object.
in	*sockfd*	Socket Descriptor.
in	*foreign_sockaddr*	Remote socket address.

References SOCKINFO::foreign_sockaddr, sock_init(), and UPNP_E_SUCCESS.

7.5.4.5 sock_make_blocking()

```
int sock_make_blocking (
            SOCKET sock )
```

Make socket blocking.

Returns

0 if successful, -1 otherwise.

Referenced by Check_Connect_And_Wait_Connection().

7.5.4.6 sock_make_no_blocking()

```
int sock_make_no_blocking (
            SOCKET sock )
```

Make socket non-blocking.

Returns

0 if successful, -1 otherwise.

Referenced by Check_Connect_And_Wait_Connection(), and create_ssdp_sock_reqv4().

7.5.4.7 sock_read()

```
int sock_read (
            SOCKINFO * info,
            char * buffer,
            size_t bufsize,
            int * timeoutSecs )
```

Reads data on socket in sockinfo.

Returns

Integer:

- `numBytes` - On Success, no of bytes received.
- `UPNP_E_TIMEDOUT` - Timeout.
- `UPNP_E_SOCKET_ERROR` - Error on socket calls.

Parameters

`in`	*info*	Socket Information Object.
`out`	*buffer*	Buffer to get data to.
`in`	*bufsize*	Size of the buffer.
`in,out`	*timeoutSecs*	timeout value.

References sock_read_write().

Referenced by http_ReadHttpResponse(), http_RecvMessage(), http_RecvPostMessage(), and ReadResponseLineAndHeaders().

7.5.4.8 sock_read_write()

```
static int sock_read_write (
```

```
            SOCKINFO * info,
            char * buffer,
            size_t bufsize,
            int * timeoutSecs,
            int bRead )  [static]
```

Receives or sends data. Also returns the time taken to receive or send data.

Returns

- numBytes - On Success, no of bytes received or sent or
- UPNP_E_TIMEDOUT - Timeout
- UPNP_E_SOCKET_ERROR - Error on socket calls

Parameters

in	*info*	Socket Information Object.
out	*buffer*	Buffer to get data to or send data from.
in	*bufsize*	Size of the buffer.
in	*timeoutSecs*	timeout value.
in	*bRead*	Boolean value specifying read or write option.

References SOCKINFO::socket, UPNP_E_SOCKET_ERROR, and UPNP_E_TIMEDOUT.

Referenced by sock_read(), and sock_write().

7.5.4.9 sock_write()

```
int sock_write (
            SOCKINFO * info,
            const char * buffer,
            size_t bufsize,
            int * timeoutSecs )
```

Writes data on the socket in sockinfo.

Returns

Integer:

- numBytes - On Success, no of bytes received.
- UPNP_E_TIMEDOUT - Timeout.
- UPNP_E_SOCKET_ERROR - Error on socket calls.

Parameters

in	*info*	Socket Information Object.
in	*buffer*	Buffer to send data from.
in	*bufsize*	Size of the buffer.
in,out	*timeoutSecs*	timeout value.

References sock_read_write().

Referenced by http_EndHttpRequest(), http_SendMessage(), and http_WriteHttpRequest().

7.5.5 Variable Documentation

7.5.5.1 foreign_sockaddr

```
struct sockaddr_storage SOCKINFO::foreign_sockaddr
```

The following two fields are filled only in incoming requests.

Referenced by gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_re
gena_validate_delivery_urls(), handle_invoke_action(), handle_query_variable(), process_request(), soap_device_callback(),
and sock_init_with_ip().

7.5.5.2 socket

```
SOCKET SOCKINFO::socket
```

Handle/descriptor to a socket.

Referenced by sock_destroy(), sock_init(), and sock_read_write().

7.6 SSDP Library

Files

- file ssdplib.h
- file ssdp_ctrlpt.c
- file ssdp_device.c
- file ssdp_server.c
- file SSDPResultDataCallback.h

  SSDPResultData object declararion.

Data Structures

- struct SsdpEventStruct
- struct TData
- struct ssdpsearchreply
- struct ssdpsearcharg
- struct ssdpsearchexparg
- struct ssdp_thread_data
- struct SSDPSockArray

Macros

- #define **BUFSIZE** (size_t)2500
- #define **SSDP_IP** "239.255.255.250"
- #define **SSDP_IPV6_LINKLOCAL** "FF02::C"
- #define **SSDP_IPV6_SITELOCAL** "FF05::C"
- #define **SSDP_PORT** 1900
- #define **NUM_TRY** 3
- #define **THREAD_LIMIT** 50
- #define **COMMAND_LEN** 300
- #define **MSGTYPE_SHUTDOWN** 0
- #define **MSGTYPE_ADVERTISEMENT** 1
- #define **MSGTYPE_REPLY** 2
- #define **MAX_TIME_TOREAD** 45

Typedefs

- typedef enum SsdpSearchType SType

Enumerations

- enum SsdpSearchType {
  SSDP_SERROR = -1 , **SSDP_ALL** , **SSDP_ROOTDEVICE** , **SSDP_DEVICEUDN** ,
  SSDP_DEVICETYPE , **SSDP_SERVICE** }
- enum **Listener** { **Idle** , **Stopping** , **Running** }

Functions

- static void send_search_result (void *data)

  Sends a callback to the control point application with a SEARCH result.
- static int CreateClientRequestPacket (char *RqstBuf, size_t RqstBufSize, int Mx, char *SearchTarget, int AddressFamily)

  Creates a HTTP search request packet depending on the input parameter.
- static void **searchExpired** (void *arg)
- static int NewRequestHandler (struct sockaddr *DestAddr, int NumPacket, char **RqPacket)

  Works as a request handler which passes the HTTP request string to multicast channel.
- static int extractIPv6address (char *url, char *address)
- static int isUrlV6UlaGua (char *descdocUrl)
- static void CreateServicePacket (int msg_type, const char *nt, char *usn, char *location, int duration, char **packet, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates a HTTP request packet. Depending on the input parameter, it either creates a service advertisement request or service shutdown request etc.
- void **RequestHandler** ()
- static void free_ssdp_event_handler_data (void *the_data)

  Frees the ssdp request.
- static UPNP_INLINE int valid_ssdp_msg (http_message_t *hmsg)

  Does some quick checking of the ssdp msg.
- static UPNP_INLINE int start_event_handler (void *Data)

  Parses the message and dispatches it to a handler which handles the ssdp request msg.
- static void ssdp_event_handler_thread (void *the_data)

  This function is a thread that handles SSDP requests.
- static int create_ssdp_sock_v4 (SOCKET *ssdpSock)
- static int create_ssdp_sock_reqv4 (SOCKET *ssdpReqSock)

  Creates the SSDP IPv4 socket to be used by the control point.
- void **SSDPResultData_Callback** (const SSDPResultData *p)

Variables

- enum SsdpSearchType **SsdpEventStruct::RequestType**
- int **SsdpEventStruct::ErrCode**
- int **SsdpEventStruct::MaxAge**
- int **SsdpEventStruct::Mx**
- char **SsdpEventStruct::UDN** [LINE_SIZE]
- char **SsdpEventStruct::DeviceType** [LINE_SIZE]
- char **SsdpEventStruct::ServiceType** [LINE_SIZE]
- char **SsdpEventStruct::Location** [LINE_SIZE]
- char **SsdpEventStruct::HostAddr** [LINE_SIZE]
- char **SsdpEventStruct::Os** [LINE_SIZE]
- char **SsdpEventStruct::Ext** [LINE_SIZE]
- char **SsdpEventStruct::Date** [LINE_SIZE]
- struct sockaddr * **SsdpEventStruct::DestAddr**
- void * **SsdpEventStruct::Cookie**
- int **TData::Mx**
- void * **TData::Cookie**
- char * **TData::Data**
- struct sockaddr_storage **TData::DestAddr**
- int **ssdpsearchreply::MaxAge**
- UpnpDevice_Handle **ssdpsearchreply::handle**
- struct sockaddr_storage **ssdpsearchreply::dest_addr**
- SsdpEvent **ssdpsearchreply::event**
- int **ssdpsearcharg::timeoutEventId**
- char * **ssdpsearcharg::searchTarget**
- void * **ssdpsearcharg::cookie**
- enum SsdpSearchType **ssdpsearcharg::requestType**
- int **ssdpsearchexparg::handle**
- int **ssdpsearchexparg::timeoutEventId**
- http_parser_t **ssdp_thread_data::parser**
- struct sockaddr_storage **ssdp_thread_data::dest_addr**
- SOCKET **gSsdpReqSocket4** = INVALID_SOCKET
- SOCKET SSDPSockArray::ssdpSock
- int SSDPSockArray::ssdpReqSock
- static const char **SERVICELIST_STR** [ ] = "serviceList"

X_USER_AGENT

can be overwritten by configure CFLAGS argument.

The {\tt X_USER_AGENT} constant specifies the value of the X-User-Agent: HTTP header. The value "redsonic" is needed for the DSM-320. See https://sourceforge.net/forum/message.php?msg↩_id=3166856 for more information

- typedef struct SsdpEventStruct SsdpEvent
- typedef void(* **SsdpFunPtr**) (SsdpEvent *)
- typedef struct TData **ThreadData**
- typedef struct ssdpsearchreply **SsdpSearchReply**
- typedef struct ssdpsearcharg **SsdpSearchArg**
- typedef struct ssdpsearchexparg **SsdpSearchExpArg**
- typedef int(* **ParserFun**) (char *, SsdpEvent *)
- SOCKET **gSsdpReqSocket4**

- #define **X_USER_AGENT** "redsonic"
- #define NO_ERROR_FOUND 0
- #define **E_REQUEST_INVALID** -3
- #define **E_RES_EXPIRED** -4
- #define **E_MEM_ALLOC** -5
- #define **E_HTTP_SYNTEX** -6
- #define **E_SOCKET** -7
- #define **RQST_TIMEOUT** 20

SSDP Server Functions

- int AdvertiseAndReply (int AdFlag, UpnpDevice_Handle Hnd, enum SsdpSearchType SearchType, struct sockaddr ∗DestAddr, char ∗DeviceType, char ∗DeviceUDN, char ∗ServiceType, int Exp)

  Sends SSDP advertisements, replies and shutdown messages.
- int unique_service_name (char ∗cmd, SsdpEvent ∗Evt)

  Fills the fields of the event structure like DeviceType, Device UDN and Service Type.
- enum SsdpSearchType ssdp_request_type1 (char ∗cmd)

  This function figures out the type of the SSDP search in the in the request.
- int ssdp_request_type (char ∗cmd, SsdpEvent ∗Evt)

  Starts filling the SSDP event structure based upon the request received.
- int readFromSSDPSocket (SOCKET socket)

  This function reads the data from the ssdp socket.
- int get_ssdp_sockets (MiniServerSockArray ∗out)

  Creates the IPv4 and IPv6 ssdp sockets required by the control point and device operation.

SSDP Control Point Functions

- void ssdp_handle_ctrlpt_msg (http_message_t ∗hmsg, struct sockaddr_storage ∗dest_addr, int timeout)

  This function handles the ssdp messages from the devices. These messages includes the search replies, advertisement of device coming alive and bye byes.
- int SearchByTarget (int Hnd, int Mx, char ∗St, void ∗Cookie)

  Creates and send the search request for a specific URL.

SSDP Device Functions

- void advertiseAndReplyThread (void ∗data)

  Wrapper function to reply the search request coming from the control point.
- void ssdp_handle_device_request (http_message_t ∗hmsg, struct sockaddr_storage ∗dest_addr)

  Handles the search request. It does the sanity checks of the request and then schedules a thread to send a random time reply (random within maximum time given by the control point to reply).
- int DeviceAdvertisement (char ∗DevType, int RootDev, char ∗Udn, char ∗Location, int Duration, int Address↩ Family, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the device advertisement request based on the input parameter, and send it to the multicast channel.
- int SendReply (struct sockaddr ∗DestAddr, char ∗DevType, int RootDev, char ∗Udn, char ∗Location, int Duration, int ByType, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the reply packet based on the input parameter, and send it to the client addesss given in its input parameter DestAddr.
- int DeviceReply (struct sockaddr ∗DestAddr, char ∗DevType, int RootDev, char ∗Udn, char ∗Location, int Duration, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the reply packet based on the input parameter, and send it to the client address given in its input parameter DestAddr.

- int ServiceAdvertisement (char *Udn, char *ServType, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the advertisement packet based on the input parameter, and send it to the multicast channel.
- int ServiceReply (struct sockaddr *DestAddr, char *ServType, char *Udn, char *Location, int Duration, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the advertisement packet based on the input parameter, and send it to the multicast channel.
- int ServiceShutdown (char *Udn, char *ServType, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates a HTTP service shutdown request packet and sends it to the multicast channel through RequestHandler.
- int DeviceShutdown (char *DevType, int RootDev, char *Udn, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates a HTTP device shutdown request packet and send it to the multicast channel through RequestHandler.

7.6.1 Detailed Description

7.6.2 Macro Definition Documentation

7.6.2.1 NO_ERROR_FOUND

```
#define NO_ERROR_FOUND 0
```

Error codes.

7.6.3 Typedef Documentation

7.6.3.1 SsdpEvent

```
typedef struct SsdpEventStruct SsdpEvent
```

Structure to store the SSDP information

7.6.3.2 SType

```
typedef enum SsdpSearchType SType
```

Enumeration to define all different types of ssdp searches

7.6.4 Enumeration Type Documentation

7.6.4.1 SsdpSearchType

```
enum SsdpSearchType
```

Enumeration to define all different types of ssdp searches

Enumerator

SSDP_SERROR	Unknown search command.

7.6.5 Function Documentation

7.6.5.1 AdvertiseAndReply()

```
int AdvertiseAndReply (
            int AdFlag,
            UpnpDevice_Handle Hnd,
            enum SsdpSearchType SearchType,
            struct sockaddr * DestAddr,
            char * DeviceType,
            char * DeviceUDN,
            char * ServiceType,
            int Exp )
```

Sends SSDP advertisements, replies and shutdown messages.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References Handle_Info::DescURL, DeviceAdvertisement(), Handle_Info::DeviceAf, Handle_Info::DeviceList, DeviceReply(), DeviceShutdown(), DOMString, GetHandleInfo(), ixmlElement_getElementsByTagName(), ixmlNode_getFirstChild(), ixmlNode_getNextSibling(), ixmlNode_getNodeName(), ixmlNode_getNodeValue(), ixmlNodeList_free(), ixmlNodeList_item(), Handle_Info::LowerDescURL, SendReply(), ServiceAdvertisement(), ServiceReply(), ServiceShutdown(), UPNP_E_INVALID_HANDLE, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by advertiseAndReplyThread(), UpnpSendAdvertisementLowPower(), and UpnpUnRegisterRootDeviceLowPower().

7.6.5.2 advertiseAndReplyThread()

```
void advertiseAndReplyThread (
            void * data )
```

Wrapper function to reply the search request coming from the control point.

References AdvertiseAndReply().

Referenced by ssdp_handle_device_request().

7.6.5.3 create_ssdp_sock_reqv4()

```
static int create_ssdp_sock_reqv4 (
            SOCKET * ssdpReqSock )  [static]
```

Creates the SSDP IPv4 socket to be used by the control point.

Returns

UPNP_E_SUCCESS on successful socket creation.

Parameters

out	*ssdpReqSock*	SSDP IPv4 request socket to be created.

References INVALID_SOCKET, sock_make_no_blocking(), UPNP_E_OUTOF_SOCKET, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by get_ssdp_sockets().

7.6.5.4 create_ssdp_sock_v4()

```
static int create_ssdp_sock_v4 (
            SOCKET * ssdpSock )  [static]
```

Parameters

ssdpSock	[] SSDP IPv4 socket to be created.

References gIF_IPV4, INVALID_SOCKET, UPNP_E_NETWORK_ERROR, UPNP_E_OUTOF_SOCKET, UPNP_E_SOCKET_BIND, UPNP_E_SOCKET_ERROR, UPNP_E_SUCCESS, UpnpCloseSocket, and UpnpPrintf().

Referenced by get_ssdp_sockets().

7.6.5.5 CreateClientRequestPacket()

```
static int CreateClientRequestPacket (
            char * RqstBuf,
            size_t RqstBufSize,
            int Mx,
            char * SearchTarget,
            int AddressFamily )  [static]
```

Creates a HTTP search request packet depending on the input parameter.

Parameters

in,out	*RqstBuf*	Output string in HTTP format.
in	*RqstBufSize*	RqstBuf size.
in	*Mx*	Search Target.
in	*SearchTarget*	Number of seconds to wait to collect all the responses.
in	*AddressFamily*	search address family.

References UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_ARGUMENT, and UPNP_E_SUCCESS.

Referenced by SearchByTarget().

7.6.5.6 CreateServicePacket()

```
static void CreateServicePacket (
            int msg_type,
```

```
            const char * nt,
            char * usn,
            char * location,
            int duration,
            char ** packet,
            int AddressFamily,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )  [static]
```

Creates a HTTP request packet. Depending on the input parameter, it either creates a service advertisement request or service shutdown request etc.

Parameters

in	*msg_type*	type of the message (Search Reply, Advertisement or Shutdown).
in	*nt*	ssdp type.
in	*usn*	unique service name (go in the HTTP Header).
in	*location*	Location URL.
in	*duration*	Service duration in sec.
out	*packet*	Output buffer filled with HTTP statement.
in	*AddressFamily*	Address family of the HTTP request.
in	*PowerState*	PowerState as defined by UPnP Low Power.
in	*SleepPeriod*	SleepPeriod as defined by UPnP Low Power.
in	*RegistrationState*	RegistrationState as defined by UPnP Low Power.

References gUpnpSdkNLSuuid, http_MakeMessage(), isUrlV6UlaGua(), and membuffer::size_inc.

Referenced by DeviceAdvertisement(), DeviceReply(), DeviceShutdown(), SendReply(), ServiceAdvertisement(), ServiceReply(), and ServiceShutdown().

7.6.5.7 DeviceAdvertisement()

```
int DeviceAdvertisement (
            char * DevType,
            int RootDev,
            char * Udn,
            char * Location,
            int Duration,
            int AddressFamily,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Creates the device advertisement request based on the input parameter, and send it to the multicast channel.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References CreateServicePacket(), gIF_INDEX, isUrlV6UlaGua(), NewRequestHandler(), UPNP_E_OUTOF_MEMORY, and UpnpPrintf().

Referenced by AdvertiseAndReply().

7.6.5.8 DeviceReply()

```
int DeviceReply (
            struct sockaddr * DestAddr,
            char * DevType,
            int RootDev,
            char * Udn,
            char * Location,
            int Duration,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Creates the reply packet based on the input parameter, and send it to the client address given in its input parameter DestAddr.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References CreateServicePacket(), NewRequestHandler(), and UPNP_E_OUTOF_MEMORY.

Referenced by AdvertiseAndReply().

7.6.5.9 DeviceShutdown()

```
int DeviceShutdown (
            char * DevType,
            int RootDev,
            char * Udn,
            char * Location,
            int Duration,
            int AddressFamily,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Creates a HTTP device shutdown request packet and send it to the multicast channel through RequestHandler.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References CreateServicePacket(), gIF_INDEX, isUrlV6UlaGua(), NewRequestHandler(), UPNP_E_OUTOF_MEMORY, and UpnpPrintf().

Referenced by AdvertiseAndReply().

7.6.5.10 extractIPv6address()

```
static int extractIPv6address (
            char * url,
            char * address )  [static]
```

Returns

1 if an inet6 @ has been found.

Referenced by isUrlV6UlaGua().

7.6.5.11 free_ssdp_event_handler_data()

```
static void free_ssdp_event_handler_data (
            void * the_data )  [static]
```

Frees the ssdp request.

Parameters

in	*the_data*	ssdp_thread_data structure. This structure contains SSDP request message.

Referenced by readFromSSDPSocket(), ssdp_event_handler_thread(), and start_event_handler().

7.6.5.12 get_ssdp_sockets()

```
int get_ssdp_sockets (
            MiniServerSockArray * out )
```

Creates the IPv4 and IPv6 ssdp sockets required by the control point and device operation.

Returns

UPNP_E_SUCCESS if successful else returns appropriate error.

References create_ssdp_sock_reqv4(), create_ssdp_sock_v4(), gIF_IPV4, gIF_IPV6, gIF_IPV6_ULA_GUA, INVALID_SOCKET, MServerSockArray::ssdpReqSock4, MServerSockArray::ssdpReqSock6, MServerSockArray::ssdpSock4, MServerSockArray::ssdpSock6, MServerSockArray::ssdpSock6UlaGua, UPNP_E_SUCCESS, and UpnpCloseSocket.

Referenced by StartMiniServer().

7.6.5.13 isUrlV6UlaGua()

```
static int isUrlV6UlaGua (
            char * descdocUrl )  [static]
```

Returns

1 if the Url contains an ULA or GUA IPv6 address, 0 otherwise.

References extractIPv6address().

Referenced by CreateServicePacket(), DeviceAdvertisement(), DeviceShutdown(), ServiceAdvertisement(), and ServiceShutdown().

7.6.5.14 NewRequestHandler()

```
static int NewRequestHandler (
            struct sockaddr * DestAddr,
            int NumPacket,
            char ** RqPacket )  [static]
```

Works as a request handler which passes the HTTP request string to multicast channel.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

Parameters

in	*DestAddr*	Ip address, to send the reply.
in	*NumPacket*	Number of packet to be sent.
in	*RqPacket*	.

References gIF_INDEX, gIF_IPV4, INVALID_SOCKET, UPNP_E_INVALID_PARAM, UPNP_E_NETWORK_ERROR, UPNP_E_OUTOF_SOCKET, UPNP_E_SOCKET_WRITE, UPNP_E_SUCCESS, UpnpCloseSocket, and UpnpPrintf().

Referenced by DeviceAdvertisement(), DeviceReply(), DeviceShutdown(), SendReply(), ServiceAdvertisement(), ServiceReply(), and ServiceShutdown().

7.6.5.15 readFromSSDPSocket()

```
int readFromSSDPSocket (
             SOCKET socket )
```

This function reads the data from the ssdp socket.

Returns

0 on success; -1 on error.

References membuffer::buf, free_ssdp_event_handler_data(), gRecvThreadPool, membuffer::length, http_message_t::msg, ssdp_event_handler_thread(), ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), and UpnpPrintf().

7.6.5.16 SearchByTarget()

```
int SearchByTarget (
             int Hnd,
             int Mx,
             char * St,
             void * Cookie )
```

Creates and send the search request for a specific URL.

This function implements the search request of the discovery phase. A M-SEARCH request is sent on the SSDP channel for both IPv4 and IPv6 addresses. The search target(ST) is required and must be one of the following:

- "ssdp:all" : Search for all devices and services.
- "ssdp:rootdevice" : Search for root devices only.
- "uuid:<device-uuid>" : Search for a particular device.
- "urn:schemas-upnp-org:device:<deviceType:v>"
- "urn:schemas-upnp-org:service:<serviceType:v>"
- "urn:<domain-name>:device:<deviceType:v>"
- "urn:<domain-name>:service:<serviceType:v>"

Returns

1 if successful else appropriate error.

References CreateClientRequestPacket(), GetHandleInfo(), gIF_INDEX, gIF_IPV4, gTimerThread, INVALID_SOCKET, REL_SEC, ssdp_request_type1(), SSDP_SERROR, Handle_Info::SsdpSearchList, TimerThreadSchedule(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_INTERNAL_ERROR, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, UpnpCloseSocket, and UpnpPrintf().

Referenced by UpnpSearchAsync().

7.6.5.17 send_search_result()

```
static void send_search_result (
            void * data )  [static]
```

Sends a callback to the control point application with a SEARCH result.

References SSDPResultData_delete().

Referenced by ssdp_handle_ctrlpt_msg().

7.6.5.18 SendReply()

```
int SendReply (
            struct sockaddr * DestAddr,
            char * DevType,
            int RootDev,
            char * Udn,
            char * Location,
            int Duration,
            int ByType,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Creates the reply packet based on the input parameter, and send it to the client addesss given in its input parameter DestAddr.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References CreateServicePacket(), NewRequestHandler(), and UPNP_E_OUTOF_MEMORY.

Referenced by AdvertiseAndReply().

7.6.5.19 ServiceAdvertisement()

```
int ServiceAdvertisement (
            char * Udn,
            char * ServType,
            char * Location,
            int Duration,
            int AddressFamily,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Creates the advertisement packet based on the input parameter, and send it to the multicast channel.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References CreateServicePacket(), gIF_INDEX, isUrlV6UlaGua(), NewRequestHandler(), UPNP_E_OUTOF_MEMORY, and UpnpPrintf().

Referenced by AdvertiseAndReply().

7.6.5.20 ServiceReply()

```
int ServiceReply (
            struct sockaddr * DestAddr,
            char * ServType,
            char * Udn,
            char * Location,
            int Duration,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Creates the advertisement packet based on the input parameter, and send it to the multicast channel.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References CreateServicePacket(), NewRequestHandler(), and UPNP_E_OUTOF_MEMORY.

Referenced by AdvertiseAndReply().

7.6.5.21 ServiceShutdown()

```
int ServiceShutdown (
            char * Udn,
            char * ServType,
            char * Location,
            int Duration,
            int AddressFamily,
            int PowerState,
            int SleepPeriod,
            int RegistrationState )
```

Creates a HTTP service shutdown request packet and sends it to the multicast channel through RequestHandler.

Returns

UPNP_E_SUCCESS if successful else appropriate error.

References CreateServicePacket(), gIF_INDEX, isUrlV6UlaGua(), NewRequestHandler(), UPNP_E_OUTOF_MEMORY, and UpnpPrintf().

Referenced by AdvertiseAndReply().

7.6.5.22 ssdp_event_handler_thread()

```
static void ssdp_event_handler_thread (
            void * the_data )  [static]
```

This function is a thread that handles SSDP requests.

Parameters

the_data	[] ssdp_thread_data structure. This structure contains SSDP request message.

References free_ssdp_event_handler_data(), http_message_t::method, http_message_t::request_method, ssdp_handle_ctrlpt_msg(), ssdp_handle_device_request(), and start_event_handler().

Referenced by readFromSSDPSocket().

7.6.5.23 ssdp_handle_ctrlpt_msg()

```
void ssdp_handle_ctrlpt_msg (
            http_message_t * hmsg,
            struct sockaddr_storage * dest_addr,
            int timeout )
```

This function handles the ssdp messages from the devices. These messages includes the search replies, advertisement of device coming alive and bye byes.

References memptr::buf, Handle_Info::Callback, Handle_Info::Cookie, GetClientHandleInfo(), GetHandleInfo(), gRecvThreadPool, http_message_t::is_request, memptr::length, PARSE_OK, send_search_result(), ssdp_request_type(), SSDPResultData_delete(), SSDPResultData_new(), SSDPResultData_set_Cookie(), SSDPResultData_set_CtrlptCallback(), SSDPResultData_set_Param(), Handle_Info::SsdpSearchList, http_message_t::status_code, ThreadPoolAdd(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), unique_service_name(), UPNP_DISCOVERY_ADVERTISEMENT_ALIVE, UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE, UPNP_DISCOVERY_SEARCH_TIMEOUT, UPNP_E_SUCCESS, UpnpDiscovery_delete(), UpnpDiscovery_get_Expires(), UpnpDiscovery_get_Location(), UpnpDiscovery_new(), UpnpDiscovery_set_DestAddr(), UpnpDiscovery_set_ErrCode(), UpnpDiscovery_set_Expires(), UpnpDiscovery_strcpy_Date(), UpnpDiscovery_strcpy_DeviceID(), UpnpDiscovery_strcpy_DeviceType(), UpnpDiscovery_strcpy_ServiceType(), UpnpDiscovery_strncpy_Ext(), UpnpDiscovery_strncpy_Location(), UpnpDiscovery_strncpy_Os(), and UpnpString_get_Length().

Referenced by ssdp_event_handler_thread().

7.6.5.24 ssdp_handle_device_request()

```
void ssdp_handle_device_request (
            http_message_t * hmsg,
            struct sockaddr_storage * dest_addr )
```

Handles the search request. It does the sanity checks of the request and then schedules a thread to send a random time reply (random within maximum time given by the control point to reply).

References advertiseAndReplyThread(), memptr::buf, GetDeviceHandleInfo(), gTimerThread, memptr::length, REL_SEC, ssdp_request_type(), TimerThreadSchedule(), TPJobInit(), TPJobSetFreeFunction(), and UpnpPrintf().

Referenced by ssdp_event_handler_thread().

7.6.5.25 ssdp_request_type()

```
int ssdp_request_type (
            char * cmd,
            SsdpEvent * Evt )
```

Starts filling the SSDP event structure based upon the request received.

Returns

0 on success; -1 on error.

References NO_ERROR_FOUND, ssdp_request_type1(), SSDP_SERROR, and unique_service_name().

Referenced by ssdp_handle_ctrlpt_msg(), and ssdp_handle_device_request().

7.6.5.26 ssdp_request_type1()

```
enum SsdpSearchType ssdp_request_type1 (
            char * cmd )
```

This function figures out the type of the SSDP search in the in the request.

Returns

enum SsdpSearchType. Returns appropriate search type, else returns SSDP_ERROR

References SSDP_SERROR.

Referenced by SearchByTarget(), and ssdp_request_type().

7.6.5.27 start_event_handler()

```
static UPNP_INLINE int start_event_handler (
            void * Data )  [static]
```

Parses the message and dispatches it to a handler which handles the ssdp request msg.

Returns

0 if successful, -1 if error.

Parameters

in	*Data*	ssdp_thread_data structure. This structure contains SSDP request message.

References free_ssdp_event_handler_data(), http_message_t::method, PARSE_FAILURE, PARSE_SUCCESS, UpnpPrintf(), valid_ssdp_msg(), and http_parser_t::valid_ssdp_notify_hack.

Referenced by ssdp_event_handler_thread().

7.6.5.28 unique_service_name()

```
int unique_service_name (
            char * cmd,
            SsdpEvent * Evt )
```

Fills the fields of the event structure like DeviceType, Device UDN and Service Type.

Returns

0 if successful else -1.

Referenced by ssdp_handle_ctrlpt_msg(), and ssdp_request_type().

7.6.5.29 valid_ssdp_msg()

```
static UPNP_INLINE int valid_ssdp_msg (
            http_message_t * hmsg )  [static]
```

Does some quick checking of the ssdp msg.

Returns

1 if msg is valid, else 0.

Parameters

in	*hmsg*	ssdp_thread_data structure. This structure contains SSDP request message.

References http_message_t::method, http_message_t::request_method, UpnpPrintf(), and http_message_t::uri.

Referenced by start_event_handler().

7.6.6 Variable Documentation

7.6.6.1 ssdpReqSock

```
int SSDPSockArray::ssdpReqSock
```

socket for sending search requests and receiving search replies

7.6.6.2 ssdpSock

```
SOCKET SSDPSockArray::ssdpSock
```

socket for incoming advertisments and search requests

7.7 XML API

Data Structures

- struct _IXML_Node

  Data structure common to all types of nodes.
- struct _IXML_Document

  Data structure representing the DOM Document.
- struct _IXML_CDATASection

  Data structure representing a CDATA section node.
- struct _IXML_Element

  Data structure representing an Element node.
- struct _IXML_ATTR

  Data structure representing an Attribute node.
- struct _IXML_Text

  Data structure representing a Text node.
- struct _IXML_NodeList

  Data structure representing a list of nodes.
- struct _IXML_NamedNodeMap

  Data structure representing a list of named nodes.

Macros

- #define DOMString char *

  The type of DOM strings.

Variables

- char * **_IXML_Node::nodeName**
- char * **_IXML_Node::nodeValue**
- IXML_NODE_TYPE **_IXML_Node::nodeType**
- char * **_IXML_Node::namespaceURI**
- char * **_IXML_Node::prefix**
- char * **_IXML_Node::localName**
- int **_IXML_Node::readOnly**
- Nodeptr **_IXML_Node::parentNode**
- Nodeptr **_IXML_Node::firstChild**
- Nodeptr **_IXML_Node::prevSibling**
- Nodeptr **_IXML_Node::nextSibling**
- Nodeptr **_IXML_Node::firstAttr**
- Docptr **_IXML_Node::ownerDocument**
- IXML_Node **_IXML_Document::n**
- IXML_Node **_IXML_CDATASection::n**
- IXML_Node **_IXML_Element::n**
- char * **_IXML_Element::tagName**
- IXML_Node **_IXML_ATTR::n**
- int **_IXML_ATTR::specified**
- IXML_Element * **_IXML_ATTR::ownerElement**
- IXML_Node **_IXML_Text::n**
- IXML_Node * **_IXML_NodeList::nodeItem**
- struct _IXML_NodeList * **_IXML_NodeList::next**
- IXML_Node * **_IXML_NamedNodeMap::nodeItem**
- struct _IXML_NamedNodeMap * **_IXML_NamedNodeMap::next**

DOM Interfaces

The Document Object Model consists of a set of objects and interfaces for accessing and manipulating documents. IXML does not implement all the interfaces documented in the DOM2-Core recommendation but defines a subset of the most useful interfaces. A description of the supported interfaces and methods is presented in this section.

For a complete discussion on the object model, the object hierarchy, etc., refer to section 1.1 of the DOM2-Core recommendation.

- enum IXML_NODE_TYPE {
  eINVALID_NODE = 0 , **eELEMENT_NODE** = 1 , **eATTRIBUTE_NODE** = 2 , **eTEXT_NODE** = 3 ,
  eCDATA_SECTION_NODE = 4 , **eENTITY_REFERENCE_NODE** = 5 , **eENTITY_NODE** = 6 , **e↩**
  PROCESSING_INSTRUCTION_NODE = 7 ,
  eCOMMENT_NODE = 8 , **eDOCUMENT_NODE** = 9 , **eDOCUMENT_TYPE_NODE** = 10 , **eDOCUMENT_↩**
  FRAGMENT_NODE = 11 ,
  eNOTATION_NODE = 12 }

  The type of the DOM node.
- enum IXML_ERRORCODE {
  IXML_SUCCESS = 0 , **IXML_INDEX_SIZE_ERR** = 1 , **IXML_DOMSTRING_SIZE_ERR** = 2 , **IXML_↩**
  HIERARCHY_REQUEST_ERR = 3 ,
  IXML_WRONG_DOCUMENT_ERR = 4 , **IXML_INVALID_CHARACTER_ERR** = 5 , **IXML_NO_DATA_↩**
  ALLOWED_ERR = 6 , **IXML_NO_MODIFICATION_ALLOWED_ERR** = 7 ,
  IXML_NOT_FOUND_ERR = 8 , **IXML_NOT_SUPPORTED_ERR** = 9 , **IXML_INUSE_ATTRIBUTE_ERR** =
  10 , **IXML_INVALID_STATE_ERR** = 11 ,
  IXML_SYNTAX_ERR = 12 , **IXML_INVALID_MODIFICATION_ERR** = 13 , **IXML_NAMESPACE_ERR** = 14 ,
  IXML_INVALID_ACCESS_ERR = 15 ,
  IXML_NO_SUCH_FILE = 101 , **IXML_INSUFFICIENT_MEMORY** = 102 , **IXML_FILE_DONE** = 104 , **IXML↩**
  _INVALID_PARAMETER = 105 ,
  IXML_FAILED = 106 , **IXML_INVALID_ITEM_NUMBER** = 107 }

  Error codes returned by the XML API, see the DOM spec.
- typedef struct _IXML_Document * **Docptr**
- typedef struct _IXML_Node * **Nodeptr**
- typedef struct _IXML_Node IXML_Node

  Data structure common to all types of nodes.
- typedef struct _IXML_Document IXML_Document

  Data structure representing the DOM Document.
- typedef struct _IXML_CDATASection IXML_CDATASection

  Data structure representing a CDATA section node.
- typedef struct _IXML_Element IXML_Element

  Data structure representing an Element node.
- typedef struct _IXML_ATTR IXML_Attr

  Data structure representing an Attribute node.
- typedef struct _IXML_Text IXML_Text

  Data structure representing a Text node.
- typedef struct _IXML_NodeList IXML_NodeList

  Data structure representing a list of nodes.
- typedef struct _IXML_NamedNodeMap IXML_NamedNodeMap

  Data structure representing a list of named nodes.
- #define **DOCUMENTNODENAME** "#document"
- #define **TEXTNODENAME** "#text"
- #define **CDATANODENAME** "#cdata-section"

Interface Node

The **Node** interface forms the primary datatype for all other DOM objects. Every other interface is derived from this interface, inheriting its functionality. For more information, refer to DOM2-Core page 34. (Note: within the IXML library the NamedNodeMap and NodeList interfaces are the only interfaces that are not DOM objects and hence do not inherit from Node)

- UPNP_EXPORT_SPEC const char * ixmlNode_getNodeName (IXML_Node *nodeptr)

  *Returns the name of the **Node**, depending on what type of **Node** it is, in a read-only string.*
- UPNP_EXPORT_SPEC const char * ixmlNode_getNodeValue (IXML_Node *nodeptr)

  *Returns the value of the **Node** as a string.*
- UPNP_EXPORT_SPEC int ixmlNode_setNodeValue (IXML_Node *nodeptr, const char *newNodeValue)

  *Assigns a new value to a **Node**.*
- UPNP_EXPORT_SPEC unsigned short ixmlNode_getNodeType (IXML_Node *nodeptr)

  *Retrieves the type of a **Node**. Note that not all possible return values are actually implemented.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getParentNode (IXML_Node *nodeptr)

  *Retrieves the parent **Node** for a **Node**.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlNode_getChildNodes (IXML_Node *nodeptr)

  *Retrieves the list of children of a **Node** in a **NodeList** structure.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getFirstChild (IXML_Node *nodeptr)

  *Retrieves the first child **Node** of a **Node**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getLastChild (IXML_Node *nodeptr)

  *Retrieves the last child **Node** of a **Node**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getPreviousSibling (IXML_Node *nodeptr)

  *Retrieves the sibling **Node** immediately preceding this **Node**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getNextSibling (IXML_Node *nodeptr)

  *Retrieves the sibling **Node** immediately following this **Node**.*
- UPNP_EXPORT_SPEC IXML_NamedNodeMap * ixmlNode_getAttributes (IXML_Node *nodeptr)

  *Retrieves the attributes of a **Node**, if it is an **Element** node, in a **NamedNodeMap** structure.*
- UPNP_EXPORT_SPEC IXML_Document * ixmlNode_getOwnerDocument (IXML_Node *nodeptr)

  *Retrieves the document object associated with this **Node**.*
- UPNP_EXPORT_SPEC const char * ixmlNode_getNamespaceURI (IXML_Node *nodeptr)

  *Retrieves the namespace URI for a **Node** as a **DOMString**.*
- UPNP_EXPORT_SPEC const char * ixmlNode_getPrefix (IXML_Node *nodeptr)

  Retrieves the namespace prefix, if present.
- UPNP_EXPORT_SPEC const char * ixmlNode_getLocalName (IXML_Node *nodeptr)

  *Retrieves the local name of a **Node**, if present.*
- UPNP_EXPORT_SPEC int ixmlNode_insertBefore (IXML_Node *nodeptr, IXML_Node *newChild, IXML_Node *refChild)

  *Inserts a new child **Node** before the existing child **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_replaceChild (IXML_Node *nodeptr, IXML_Node *newChild, IXML_Node *oldChild, IXML_Node **returnNode)

  *Replaces an existing child **Node** with a new child **Node** in the list of children of a **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_removeChild (IXML_Node *nodeptr, IXML_Node *oldChild, IXML_Node **returnNode)

  *Removes a child from the list of children of a **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_appendChild (IXML_Node *nodeptr, IXML_Node *newChild)

  *Appends a child **Node** to the list of children of a **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_hasChildNodes (IXML_Node *nodeptr)

  *Queries whether or not a **Node** has children.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_cloneNode (IXML_Node *nodeptr, int deep)

*Clones a **Node**.*

- UPNP_EXPORT_SPEC int ixmlNode_hasAttributes (IXML_Node *nodeptr)

  *Queries whether this **Node** has attributes.*
- UPNP_EXPORT_SPEC void ixmlNode_free (IXML_Node *nodeptr)

  *Frees a **Node** and all **Nodes** in its subtree.*

Interface Attr

The **Attr** interface represents an attribute of an **Element**. The document type definition (DTD) or schema usually dictate the allowable attributes and values for a particular element.

For more information, refer to the *Interface Attr* section in the DOM2-Core.

- UPNP_EXPORT_SPEC void ixmlAttr_free (IXML_Attr *attrNode)

  *Frees an **Attr** node.*

Interface CDATASection

The **CDATASection** is used to escape blocks of text containing characters that would otherwise be regarded as markup. CDATA sections cannot be nested. Their primary purpose is for including material such XML fragments, without needing to escape all the delimiters.

For more information, refer to the *Interface CDATASection* section in the DOM2-Core.

- UPNP_EXPORT_SPEC void ixmlCDATASection_init (IXML_CDATASection *nodeptr)

  *Initializes a **CDATASection** node.*
- UPNP_EXPORT_SPEC void ixmlCDATASection_free (IXML_CDATASection *nodeptr)

  *Frees a **CDATASection** node.*

Interface Document

The **Document** interface represents the entire XML document. In essence, it is the root of the document tree and provides the primary interface to the elements of the document.

For more information, refer to the *Interface Document* section in the DOM2Core.

- UPNP_EXPORT_SPEC void ixmlDocument_init (IXML_Document *nodeptr)

  *Initializes a **Document** node.*
- UPNP_EXPORT_SPEC int ixmlDocument_createDocumentEx (IXML_Document **doc)

  *Creates a new empty **Document** node.*
- UPNP_EXPORT_SPEC IXML_Document * ixmlDocument_createDocument (void)

  *Creates a new empty **Document** node.*
- UPNP_EXPORT_SPEC int ixmlDocument_createElementEx (IXML_Document *doc, const char *tagName, IXML_Element **rtElement)

  *Creates a new **Element** node with the given tag name.*
- UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_createElement (IXML_Document *doc, const char *tagName)

  *Creates a new **Element** node with the given tag name.*

- UPNP_EXPORT_SPEC int ixmlDocument_createTextNodeEx (IXML_Document *doc, const char *data, IXML_Node **textNode)

  *Creates a new **Text** node with the given data.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlDocument_createTextNode (IXML_Document *doc, const char *data)

  *Creates a new **Text** node with the given data.*
- UPNP_EXPORT_SPEC int ixmlDocument_createCDATASectionEx (IXML_Document *doc, const char *data, IXML_CDATASection **cdNode)

  *Creates a new **CDATASection** node with given data.*
- UPNP_EXPORT_SPEC IXML_CDATASection * ixmlDocument_createCDATASection (IXML_Document *doc, const char *data)

  *Creates a new **CDATASection** node with given data.*
- UPNP_EXPORT_SPEC IXML_Attr * ixmlDocument_createAttribute (IXML_Document *doc, const char *name)

  *Creates a new **Attr** node with the given name.*
- UPNP_EXPORT_SPEC int ixmlDocument_createAttributeEx (IXML_Document *doc, const char *name, IXML_Attr **attrNode)

  *Creates a new **Attr** node with the given name.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlDocument_getElementsByTagName (IXML_Document *doc, const char *tagName)

  *Returns a **NodeList** of all **Elements** that match the given tag name in the order in which they were encountered in a preorder traversal of the **Document** tree.*
- UPNP_EXPORT_SPEC int ixmlDocument_createElementNSEx (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName, IXML_Element **rtElement)

  *Creates a new **Element** node in the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_createElementNS (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName)

  *Creates a new **Element** node in the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC int ixmlDocument_createAttributeNSEx (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName, IXML_Attr **attrNode)

  *Creates a new **Attr** node with the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC IXML_Attr * ixmlDocument_createAttributeNS (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName)

  *Creates a new **Attribute** node with the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlDocument_getElementsByTagNameNS (IXML_Document *doc, const char *namespaceURI, const char *localName)

  *Returns a **NodeList** of **Elements** that match the given local name and namespace URI in the order they are encountered in a preorder traversal of the **Document** tree.*
- UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_getElementById (IXML_Document *doc, const char *tagName)

  *Returns the **Element** whose `ID` matches that given id.*
- UPNP_EXPORT_SPEC void ixmlDocument_free (IXML_Document *doc)

  *Frees a **Document** object and all **Nodes** associated with it.*
- UPNP_EXPORT_SPEC int ixmlDocument_importNode (IXML_Document *doc, IXML_Node *importNode, int deep, IXML_Node **rtNode)

  *Imports a **Node** from another **Document** into this **Document**.*

Interface Element

The **Element** interface represents an element in an XML document. Only **Elements** are allowed to have attributes, which are stored in the `attributes` member of a **Node**. The **Element** interface extends the **Node** interface and adds more operations to manipulate attributes.

- UPNP_EXPORT_SPEC void ixmlElement_init (IXML_Element *element)

  *Initializes a **IXML_Element** node.*
- UPNP_EXPORT_SPEC const char * ixmlElement_getTagName (IXML_Element *element)

  Returns the name of the tag as a constant string.
- UPNP_EXPORT_SPEC const char * ixmlElement_getAttribute (IXML_Element *element, const char *name)

  *Retrieves an attribute of an **Element** by name.*
- UPNP_EXPORT_SPEC int ixmlElement_setAttribute (IXML_Element *element, const char *name, const char *value)

  *Adds a new attribute to an **Element**.*
- UPNP_EXPORT_SPEC int ixmlElement_removeAttribute (IXML_Element *element, const char *name)

  Removes an attribute value by name. The attribute node is not removed.
- UPNP_EXPORT_SPEC IXML_Attr * ixmlElement_getAttributeNode (IXML_Element *element, const char *name)

  *Retrieves an attribute node by name. See **ixmlElement_getAttributeNodeNS** to retrieve an attribute node using a qualified name or namespace URI.*
- UPNP_EXPORT_SPEC int ixmlElement_setAttributeNode (IXML_Element *element, IXML_Attr *newAttr, IXML_Attr **rtAttr)

  *Adds a new attribute node to an **Element**.*
- UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNode (IXML_Element *element, IXML_Attr *old↩ Attr, IXML_Attr **rtAttr)

  *Removes the specified attribute node from an **Element**.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlElement_getElementsByTagName (IXML_Element *element, const char *tagName)

  *Returns a **NodeList** of all descendant **Elements** with a given tag name, in the order in which they are encountered in a pre-order traversal of this **Element** tree.*
- UPNP_EXPORT_SPEC const char * ixmlElement_getAttributeNS (IXML_Element *element, const char *namespaceURI, const char *localname)

  Retrieves an attribute value using the local name and namespace URI.
- UPNP_EXPORT_SPEC int ixmlElement_setAttributeNS (IXML_Element *element, const char *namespace↩ URI, const char *qualifiedName, const char *value)

  *Adds a new attribute to an **Element** using the local name and namespace URI.*
- UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNS (IXML_Element *element, const char *namespaceURI, const char *localName)

  Removes an attribute using the namespace URI and local name.
- UPNP_EXPORT_SPEC IXML_Attr * ixmlElement_getAttributeNodeNS (IXML_Element *element, const char *namespaceURI, const char *localName)

  *Retrieves an **Attr** node by local name and namespace URI.*
- UPNP_EXPORT_SPEC int ixmlElement_setAttributeNodeNS (IXML_Element *element, IXML_Attr *newAttr, IXML_Attr **rcAttr)

  Adds a new attribute node to the element node specified.
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlElement_getElementsByTagNameNS (IXML_Element *element, const char *namespaceURI, const char *localName)

  *Returns a **NodeList** of all descendant **Elements** with a given local name and namespace in the order in which they are encountered in the pre-order traversal of the **Element** tree.*
- UPNP_EXPORT_SPEC int ixmlElement_hasAttribute (IXML_Element *element, const char *name)

  *Queries whether the **Element** has an attribute with the given name or a default value.*
- UPNP_EXPORT_SPEC int ixmlElement_hasAttributeNS (IXML_Element *element, const char *namespace↩ URI, const char *localName)

  *Queries whether the **Element** has an attribute with the given local name and namespace URI or has a default value for that attribute.*
- UPNP_EXPORT_SPEC void ixmlElement_free (IXML_Element *element)

  *Frees the given **Element** and any subtree of the **Element**.*

Interface NamedNodeMap

A **NamedNodeMap** object represents a list of objects that can be accessed by name. A **NamedNodeMap** maintains the objects in no particular order. The **Node** interface uses a **NamedNodeMap** to maintain the attributes of a node.

- UPNP_EXPORT_SPEC unsigned long ixmlNamedNodeMap_getLength (IXML_NamedNodeMap *nnMap)

  *Returns the number of items contained in this **NamedNodeMap**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_getNamedItem (IXML_NamedNodeMap *nn↩ Map, const char *name)

  *Retrieves a **Node** from the **NamedNodeMap** by name.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_setNamedItem (IXML_NamedNodeMap *nn↩ Map, IXML_Node *arg)

  *Adds a new **Node** to the **NamedNodeMap** using the **Node** name attribute.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_removeNamedItem (IXML_NamedNodeMap *nnMap, const char *name)

  *Removes a **Node** from a **NamedNodeMap** specified by name.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_item (IXML_NamedNodeMap *nnMap, unsigned long index)

  *Retrieves the indexth item in the map. If index is greater than or equal to the number of nodes in this map, this returns **NULL**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_getNamedItemNS (IXML_NamedNodeMap *nnMap, const char **namespaceURI, const char *localName)

  *Retrieves a **Node** from a **NamedNodeMap** specified by namespace URI and local name.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_setNamedItemNS (IXML_NamedNodeMap *nnMap, IXML_Node *arg)

  *Adds a new **Node** to the **NamedNodeMap** using the **Node** local name and namespace URI attributes.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_removeNamedItemNS (IXML_NamedNodeMap *nnMap, const char *namespaceURI, const char *localName)

  *Removes a **Node** from a **NamedNodeMap** specified by namespace URI and local name.*
- UPNP_EXPORT_SPEC void ixmlNamedNodeMap_free (IXML_NamedNodeMap *nnMap)

  *Frees a **NamedNodeMap**.*

Interface NodeList

The **NodeList** interface abstracts an ordered collection of nodes. Note that changes to the underlying nodes will change the nodes contained in a **NodeList**. The DOM2-Core refers to this as being *live*.

- UPNP_EXPORT_SPEC IXML_Node * ixmlNodeList_item (IXML_NodeList *nList, unsigned long index)

  *Retrieves a **Node** from a **NodeList** specified by a numerical index.*
- UPNP_EXPORT_SPEC unsigned long ixmlNodeList_length (IXML_NodeList *nList)

  *Returns the number of **Nodes** in a **NodeList**.*
- UPNP_EXPORT_SPEC void ixmlNodeList_free (IXML_NodeList *nList)

  *Frees a **NodeList** object.*

IXML API

The IXML API contains utility functions that are not part of the standard DOM interfaces. They include functions to create a DOM structure from a file or buffer, create an XML file from a DOM structure, and manipulate DOMString objects.

- UPNP_EXPORT_SPEC char * ixmlPrintDocument (IXML_Document *doc)

  *Renders a **Node** and all sub-elements into an XML document representation.*
- UPNP_EXPORT_SPEC char * ixmlPrintNode (IXML_Node *doc)

  *Renders a **Node** and all sub-elements into an XML text representation.*
- UPNP_EXPORT_SPEC char * ixmlDocumenttoString (IXML_Document *doc)

  *Renders a **Node** and all sub-elements into an XML document representation.*
- UPNP_EXPORT_SPEC char * ixmlNodetoString (IXML_Node *doc)

  *Renders a **Node** and all sub-elements into an XML text representation. The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.*
- UPNP_EXPORT_SPEC void ixmlRelaxParser (char errorChar)

  Makes the XML parser more tolerant to malformed text.
- UPNP_EXPORT_SPEC IXML_Document * ixmlParseBuffer (const char *buffer)

  Parses an XML text buffer converting it into an IXML DOM representation.
- UPNP_EXPORT_SPEC int ixmlParseBufferEx (const char *buffer, IXML_Document **doc)

  Parses an XML text buffer converting it into an IXML DOM representation.
- UPNP_EXPORT_SPEC IXML_Document * ixmlLoadDocument (const char *xmlFile)

  Parses an XML text file converting it into an IXML DOM representation.
- UPNP_EXPORT_SPEC int ixmlLoadDocumentEx (const char *xmlFile, IXML_Document **doc)

  Parses an XML text file converting it into an IXML DOM representation.
- UPNP_EXPORT_SPEC char * ixmlCloneDOMString (const char *src)

  *Clones an existing **DOMString**.*
- UPNP_EXPORT_SPEC void ixmlFreeDOMString (char *buf)

  *Frees a **DOMString**.*

7.7.1 Detailed Description

7.7.2 Macro Definition Documentation

7.7.2.1 DOMString

```
#define DOMString char *
```

The type of DOM strings.

7.7.3 Typedef Documentation

7.7.3.1 IXML_Attr

```
typedef struct _IXML_ATTR IXML_Attr
```

Data structure representing an Attribute node.

7.7.3.2 IXML_CDATASection

```
typedef struct _IXML_CDATASection IXML_CDATASection
```

Data structure representing a CDATA section node.

7.7.3.3 IXML_Document

```
typedef struct _IXML_Document IXML_Document
```

Data structure representing the DOM Document.

7.7.3.4 IXML_Element

```
typedef struct _IXML_Element IXML_Element
```

Data structure representing an Element node.

7.7.3.5 IXML_NamedNodeMap

```
typedef struct _IXML_NamedNodeMap IXML_NamedNodeMap
```

Data structure representing a list of named nodes.

7.7.3.6 IXML_Node

```
typedef struct _IXML_Node IXML_Node
```

Data structure common to all types of nodes.

7.7.3.7 IXML_NodeList

```
typedef struct _IXML_NodeList IXML_NodeList
```

Data structure representing a list of nodes.

7.7.3.8 IXML_Text

```
typedef struct _IXML_Text IXML_Text
```

Data structure representing a Text node.

7.7.4 Enumeration Type Documentation

7.7.4.1 IXML_ERRORCODE

```
enum IXML_ERRORCODE
```

Error codes returned by the XML API, see the DOM spec.

7.7.4.2 IXML_NODE_TYPE

```
enum IXML_NODE_TYPE
```

The type of the DOM node.

7.7.5 Function Documentation

7.7.5.1 ixmlAttr_free()

```
UPNP_EXPORT_SPEC void ixmlAttr_free (
            IXML_Attr * attrNode )
```

Frees an **Attr** node.

Parameters

attrNode	The **Attr** node to free.

References ixmlNode_free().

Referenced by ixmlElement_setAttributeNode(), ixmlNode_cloneAttr(), and Parser_processAttributeName().

7.7.5.2 ixmlCDATASection_free()

```
UPNP_EXPORT_SPEC void ixmlCDATASection_free (
            IXML_CDATASection * nodeptr )
```

Frees a **CDATASection** node.

Parameters

nodeptr	The **CDATASection** node to free.

References ixmlNode_free().

Referenced by ixmlNode_cloneCDATASect().

7.7.5.3 ixmlCDATASection_init()

```
UPNP_EXPORT_SPEC void ixmlCDATASection_init (
            IXML_CDATASection * nodeptr )
```

Initializes a **CDATASection** node.

Parameters

in	*nodeptr*	The **CDATA Section Node** to iniatialize.

Referenced by ixmlNode_cloneCDATASect().

7.7.5.4 ixmlCloneDOMString()

```
UPNP_EXPORT_SPEC char * ixmlCloneDOMString (
            const char * src )
```

Clones an existing **DOMString**.

Returns

A new **DOMString** that is a duplicate of the original or NULL if the operation could not be completed.

Parameters

in	*src*	The source **DOMString** to clone.

Referenced by addServiceTable(), GeneratePropertySet(), getElementValue(), getServiceTable(), TestClass_set_TheDomString(), UpnpExtraHeaders_set_resp(), UpnpFileInfo_set_ContentType(), UpnpStateVarComplete_set_CurrentVal(), and UpnpStateVarRequest_set_CurrentVal().

7.7.5.5 ixmlDocument_createAttribute()

```
UPNP_EXPORT_SPEC IXML_Attr * ixmlDocument_createAttribute (
            IXML_Document * doc,
            const char * name )
```

Creates a new **Attr** node with the given name.

Returns

A pointer to the new **Attr** object with the nodeName attribute set to the given name, and the localName, prefix and namespaceURI set to NULL or NULL on failure.

The value of the attribute is the empty string.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*name*	The name of the new attribute.

7.7.5.6 ixmlDocument_createAttributeEx()

```
UPNP_EXPORT_SPEC int ixmlDocument_createAttributeEx (
            IXML_Document * doc,
            const char * name,
            IXML_Attr ** attrNode )
```

Creates a new **Attr** node with the given name.

The **ixmlDocument_createAttributeEx** API differs from the **ixmlDocument_createAttribute** API in that it returns an error code describing the reason for failure rather than just NULL.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **doc** or **name** is NULL.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*name*	The name of the new attribute.
out	*attrNode*	A pointer to a **Attr** where the new object will be stored.

Referenced by Parser_processAttributeName().

7.7.5.7 ixmlDocument_createAttributeNS()

```
UPNP_EXPORT_SPEC IXML_Attr * ixmlDocument_createAttributeNS (
            IXML_Document * doc,
            const char * namespaceURI,
            const char * qualifiedName )
```

Creates a new **Attribute** node with the given qualified name and namespace URI.

Returns

A pointer to the new **Attr** node with the given namespaceURI and qualifiedName. The prefix and localname are extracted from the qualifiedName. The node value is empty. Or NULL on failure.

Parameters

in	*doc*	The owner **Document** of the new **Attribute**.
in	*namespaceURI*	The namespace URI for the attribute.
in	*qualifiedName*	The qualified name of the attribute.

7.7.5.8 ixmlDocument_createAttributeNSEx()

```
UPNP_EXPORT_SPEC int ixmlDocument_createAttributeNSEx (
            IXML_Document * doc,
            const char * namespaceURI,
            const char * qualifiedName,
            IXML_Attr ** attrNode )
```

Creates a new **Attr** node with the given qualified name and namespace URI.

The **ixmlDocument_createAttributeNSEx** API differs from the **ixmlDocument_createAttributeNS** API in that it returns an error code describing the reason for failure rather than just NULL.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **doc**, **namespaceURI**, or **qualifiedName** is NULL.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters

in	*doc*	The owner **Document** of the new **Attr**.
in	*namespaceURI*	The namespace URI for the attribute.
in	*qualifiedName*	The qualified name of the attribute.
out	*attrNode*	A pointer to an **Attr** where the new object will be stored.

7.7.5.9 ixmlDocument_createCDATASection()

```
UPNP_EXPORT_SPEC IXML_CDATASection * ixmlDocument_createCDATASection (
            IXML_Document * doc,
            const char * data )
```

Creates a new **CDATASection** node with given data.

Returns

A pointer to the new **CDATASection** or NULL on failure.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*data*	The data to associate with the new **CDATASection** node.

7.7.5.10 ixmlDocument_createCDATASectionEx()

```
UPNP_EXPORT_SPEC int ixmlDocument_createCDATASectionEx (
            IXML_Document * doc,
            const char * data,
            IXML_CDATASection ** cdNode )
```

Creates a new **CDATASection** node with given data.

The **ixmlDocument_createCDATASectionEx** API differs from the **ixmlDocument_createCDATASection** API in that it returns an error code describing the reason for failure rather than just NULL.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **doc** or **data** is NULL.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*data*	The data to associate with the new **CDATASection** node.
out	*cdNode*	A pointer to a **Node** where the new object will be stored.

Referenced by Parser_parseDocument().

7.7.5.11 ixmlDocument_createDocument()

```
UPNP_EXPORT_SPEC IXML_Document * ixmlDocument_createDocument (
            void  )
```

Creates a new empty **Document** node.

Returns

A pointer to the new **Document** object with the nodeName set to "#document" or NULL on failure.

References ixmlDocument_createDocumentEx().

7.7.5.12 ixmlDocument_createDocumentEx()

```
UPNP_EXPORT_SPEC int ixmlDocument_createDocumentEx (
            IXML_Document ** doc )
```

Creates a new empty **Document** node.

The **ixmlDocument_createDocumentEx** API differs from the **ixmlDocument_createDocument** API in that it returns an error code describing the reason for the failure rather than just NULL.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters

out	*doc*	Pointer to a **Document** where the new object will be stored.

References ixmlDocument_free(), and ixmlDocument_init().

Referenced by ixmlDocument_createDocument(), and Parser_parseDocument().

7.7.5.13 ixmlDocument_createElement()

```
UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_createElement (
            IXML_Document * doc,
            const char * tagName )
```

Creates a new **Element** node with the given tag name.

The new **Element** node has a `nodeName` of **tagName** and the `localName`, `prefix`, and `namespaceURI` set to `NULL`. To create an **Element** with a namespace, see **ixmlDocument_createElementNS**.

Returns

A pointer to the new **Element** object with the node name set to tagName, and localName, prefix and namespaceURI set to `NULL`, or `NULL` on failure.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*tagName*	The tag name of the new **Element** node (case-sensitive).

Referenced by addToAction(), makeAction(), UpnpAddToPropertySet(), and UpnpCreatePropertySet().

7.7.5.14 ixmlDocument_createElementEx()

```
UPNP_EXPORT_SPEC int ixmlDocument_createElementEx (
            IXML_Document * doc,
            const char * tagName,
            IXML_Element ** rtElement )
```

Creates a new **Element** node with the given tag name.

The new **Element** node has a `nodeName` of **tagName** and the `localName`, `prefix`, and `namespaceURI` set to `NULL`. To create an **Element** with a namespace, see **ixmlDocument_createElementNS**.

The **ixmlDocument_createElementEx** API differs from the **ixmlDocument_createElement** API in that it returns an error code describing the reason for failure rather than just `NULL`.

Returns

An integer representing one of the following:

- `IXML_SUCCESS`: The operation completed successfully.
- `IXML_INVALID_PARAMETER`: Either **doc** or **tagName** is `NULL`.
- `IXML_INSUFFICIENT_MEMORY`: Not enough free memory exists to complete this operation.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*tagName*	The tag name of the new **Element** node.
out	*rtElement*	Pointer to an **Element** where the new object will be stored.

Referenced by Parser_processElementName().

7.7.5.15 ixmlDocument_createElementNS()

```
UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_createElementNS (
            IXML_Document * doc,
            const char * namespaceURI,
            const char * qualifiedName )
```

Creates a new **Element** node in the given qualified name and namespace URI.

Returns

A pointer to the new **Element** object with tagName qualifiedName, prefix and localName extraced from qualfiedName, nodeName of qualfiedName, namespaceURI of namespaceURI or NULL on failure.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*namespaceURI*	The namespace URI for the new **Element**.
in	*qualifiedName*	The qualified name of the new **Element**.

7.7.5.16 ixmlDocument_createElementNSEx()

```
UPNP_EXPORT_SPEC int ixmlDocument_createElementNSEx (
            IXML_Document * doc,
            const char * namespaceURI,
            const char * qualifiedName,
            IXML_Element ** rtElement )
```

Creates a new **Element** node in the given qualified name and namespace URI.

The **ixmlDocument_createElementNSEx** API differs from the **ixmlDocument_createElementNS** API in that it returns an error code describing the reason for failure rather than just NULL.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **doc**, **namespaceURI**, or **qualifiedName** is NULL.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*namespaceURI*	The namespace URI for the new **Element**.
in	*qualifiedName*	The qualified name of the new **Element**.
out	*rtElement*	A pointer to an **Element** where the new object will be stored.

7.7.5.17 ixmlDocument_createTextNode()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlDocument_createTextNode (
            IXML_Document * doc,
            const char * data )
```

Creates a new **Text** node with the given data.

Returns

A pointer to the new **Node** or NULL on failure.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*data*	The data to associate with the new **Text** node. It is stored in the nodeValue field.

Referenced by addToAction(), makeAction(), UpnpAddToPropertySet(), and UpnpCreatePropertySet().

7.7.5.18 ixmlDocument_createTextNodeEx()

```
UPNP_EXPORT_SPEC int ixmlDocument_createTextNodeEx (
            IXML_Document * doc,
            const char * data,
            IXML_Node ** textNode )
```

Creates a new **Text** node with the given data.

The **ixmlDocument_createTextNodeEx()** API differs from the **ixmlDocument_createTextNode** API in that it returns an error code describing the reason for failure rather than just NULL.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **doc** or **data** is NULL.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters

in	*doc*	The owner **Document** of the new node.
in	*data*	The data to associate with the new **Text** node. It is stored in nodeValue field.
out	*textNode*	A pointer to a **Node** where the new object will be stored.

Referenced by Parser_parseDocument().

7.7.5.19 ixmlDocument_free()

```
UPNP_EXPORT_SPEC void ixmlDocument_free (
            IXML_Document * doc )
```

Frees a **Document** object and all **Nodes** associated with it.

Any **Nodes** extracted via any other interface function, e.g. **ixmlDocument_GetElementById**, become invalid after this call unless explicitly cloned.

Parameters

in	*doc*	The **Document** to free.

References ixmlNode_free().

Referenced by free_action_arg(), gena_process_notification_event(), handle_invoke_action(), ixmlDocument_createDocumentEx(), ixmlNode_newDoc(), Parser_parseDocument(), soap_device_callback(), TvDeviceStateTableInit(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), UpnpSendActionExAsync(), UpnpThreadDistribution(), and UpnpUnRegisterRootDeviceLowPower().

7.7.5.20 ixmlDocument_getElementById()

```
UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_getElementById (
            IXML_Document * doc,
            const char * tagName )
```

Returns the **Element** whose `ID` matches that given id.

Returns

A pointer to the matching **Element** or `NULL` on an error.

Parameters

in	*doc*	The owner **Document** of the **Element**.
in	*tagName*	The name of the **Element**.

7.7.5.21 ixmlDocument_getElementsByTagName()

```
UPNP_EXPORT_SPEC IXML_NodeList * ixmlDocument_getElementsByTagName (
            IXML_Document * doc,
            const char * tagName )
```

Returns a **NodeList** of all **Elements** that match the given tag name in the order in which they were encountered in a preorder traversal of the **Document** tree.

Returns

A pointer to a **NodeList** containing the matching items or NULL on an error.

Parameters

in	*doc*	The **Document** to search.
in	*tagName*	The tag name to find. The special value "*" matches all tags.

Referenced by SampleUtil_GetFirstDocumentItem(), SampleUtil_GetFirstServiceList(), TvStateUpdate(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

7.7.5.22 ixmlDocument_getElementsByTagNameNS()

```
UPNP_EXPORT_SPEC IXML_NodeList * ixmlDocument_getElementsByTagNameNS (
            IXML_Document * doc,
            const char * namespaceURI,
            const char * localName )
```

Returns a **NodeList** of **Elements** that match the given local name and namespace URI in the order they are encountered in a preorder traversal of the **Document** tree.

Either **namespaceURI** or **localName** can be the special "*" character, which matches any namespace or any local name respectively.

Returns

A pointer to a **NodeList** containing the matching items or NULL on an error.

Parameters

in	*doc*	The **Document** to search.
in	*namespaceURI*	The namespace of the elements to find or "*" to match any namespace.
in	*localName*	The local name of the elements to find or "*" to match any local name.

7.7.5.23 ixmlDocument_importNode()

```
UPNP_EXPORT_SPEC int ixmlDocument_importNode (
            IXML_Document * doc,
            IXML_Node * importNode,
            int deep,
            IXML_Node ** rtNode )
```

Imports a **Node** from another **Document** into this **Document**.

The returned new **Node** does not a have parent node (parentNode is null): it is a clone of the original **Node** with the ownerDocument set to **doc**. The source node is not altered or removed from the original document.

For all nodes, importing a node creates a node object owned by the importing document, with attribute values identical to the source node's nodeName and nodeType, plus the attributes related to namespaces (prefix, local↩ Name, and namespaceURI).

As in the cloneNode operation on a node, the source node is not altered.

The **deep** parameter controls whether all the children of the **Node** are imported.

Refer to the DOM2-Core recommendation for details on importing specific node types.

Returns

An integer representing one of the following:

- `IXML_SUCCESS`: The operation completed successfully.
- `IXML_INVALID_PARAMETER`: Either **doc** or **importNode** is not a valid pointer.
- `IXML_NOT_SUPPORTED_ERR`: **importNode** is a **Document**, which cannot be imported.
- `IXML_FAILED`: The import operation failed because the **Node** to be imported could not be cloned.

Parameters

`in`	*doc*	The **Document** into which to import.
`in`	*importNode*	The **Node** to import.
`in`	*deep*	`1` to import all children of **importNode** or `0` to import only the root node.
`out`	*rtNode*	A pointer to a new **Node** owned by **doc**.

References ixmlDocument_setOwnerDocument(), ixmlNode_cloneNode(), and ixmlNode_getNodeType().

7.7.5.24 ixmlDocument_init()

```
UPNP_EXPORT_SPEC void ixmlDocument_init (
             IXML_Document * nodeptr )
```

Initializes a **Document** node.

Parameters

`in`	*nodeptr*	The **Document** node to initialize.

Referenced by ixmlDocument_createDocumentEx(), and ixmlNode_newDoc().

7.7.5.25 ixmlDocumenttoString()

```
UPNP_EXPORT_SPEC char * ixmlDocumenttoString (
             IXML_Document * doc )
```

Renders a **Node** and all sub-elements into an XML document representation.

The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.

Note that this function can be used for any **Node-derived** interface. The difference between **ixmlDocumentto↩String** and **ixmlNodetoString** is **ixmlDocumenttoString** includes the XML prolog while **ixmlNodetoString** only produces XML elements. An XML document is not well formed unless it includes the prolog and at least one element.

Returns

A **DOMString** with the XML text representation of the DOM tree or NULL on an error.

Parameters

in	*doc*	The root of the **Node** tree to render to XML text.

References ixml_membuf_append_str(), ixml_membuf_init(), and ixmlDomTreetoString().

7.7.5.26 ixmlElement_free()

```
UPNP_EXPORT_SPEC void ixmlElement_free (
            IXML_Element * element )
```

Frees the given **Element** and any subtree of the **Element**.

Parameters

in	*element*	The **Element** to free.

References ixmlNode_free().

Referenced by ixmlNode_cloneElement(), and Parser_processElementName().

7.7.5.27 ixmlElement_getAttribute()

```
UPNP_EXPORT_SPEC const char * ixmlElement_getAttribute (
            IXML_Element * element,
            const char * name )
```

Retrieves an attribute of an **Element** by name.

Returns

The value of the attribute, or **NULL** if that attribute does not have a specified value.

Parameters

in	*element*	The **Element** from which to retrieve the attribute.
in	*name*	The name of the attribute to retrieve.

7.7.5.28 ixmlElement_getAttributeNode()

```
UPNP_EXPORT_SPEC IXML_Attr * ixmlElement_getAttributeNode (
            IXML_Element * element,
            const char * name )
```

Retrieves an attribute node by name. See **ixmlElement_getAttributeNodeNS** to retrieve an attribute node using a qualified name or namespace URI.

Returns

A pointer to the attribute matching **name** or NULL on if there is no such attribute.

Parameters

in	*element*	The **Element** from which to get the attribute node.
in	*name*	The name of the attribute node to find.

7.7.5.29 ixmlElement_getAttributeNodeNS()

```
UPNP_EXPORT_SPEC IXML_Attr * ixmlElement_getAttributeNodeNS (
            IXML_Element * element,
            const char * namespaceURI,
            const char * localName )
```

Retrieves an **Attr** node by local name and namespace URI.

Returns

A pointer to an **Attribute** node with the specified attribute local name and namespace URI or NULL if there is no such attribute.

Parameters

in	*element*	The **Element** from which to get the attribute.
in	*namespaceURI*	The namespace URI of the attribute.
in	*localName*	The local name of the attribute.

7.7.5.30 ixmlElement_getAttributeNS()

```
UPNP_EXPORT_SPEC const char * ixmlElement_getAttributeNS (
            IXML_Element * element,
            const char * namespaceURI,
            const char * localname )
```

Retrieves an attribute value using the local name and namespace URI.

Returns

A **DOMString** representing the value of the matching attribute, or **NULL** if that attribute does not have the specified value.

Parameters

in	*element*	The **Element** from which to get the attribute value.
in	*namespaceURI*	The namespace URI of the attribute.
in	*localname*	The local name of the attribute.

7.7.5.31 ixmlElement_getElementsByTagName()

```
UPNP_EXPORT_SPEC IXML_NodeList * ixmlElement_getElementsByTagName (
            IXML_Element * element,
            const char * tagName )
```

Returns a **NodeList** of all *descendant* **Elements** with a given tag name, in the order in which they are encountered in a pre-order traversal of this **Element** tree.

Returns

A **NodeList** of the matching **Elements** or NULL on an error.

Parameters

in	*element*	The **Element** from which to start the search.
in	*tagName*	The name of the tag for which to search.

Referenced by AdvertiseAndReply(), removeServiceTable(), SampleUtil_GetFirstElementItem(), SampleUtil_GetFirstServiceList(), and TvStateUpdate().

7.7.5.32 ixmlElement_getElementsByTagNameNS()

```
UPNP_EXPORT_SPEC IXML_NodeList * ixmlElement_getElementsByTagNameNS (
            IXML_Element * element,
            const char * namespaceURI,
            const char * localName )
```

Returns a **NodeList** of all *descendant* **Elements** with a given local name and namespace in the order in which they are encountered in the pre-order traversal of the **Element** tree.

Returns

A **NodeList** of matching **Elements** or NULL on an error.

Parameters

in	*element*	The **Element** from which to start the search.
in	*namespaceURI*	The namespace URI of the **Elements** to find. The special value "*" matches all namespaces.
in	*localName*	The local name of the **Elements** to find. The special value "*" matches all local names.

7.7.5.33 ixmlElement_getTagName()

```
UPNP_EXPORT_SPEC const char * ixmlElement_getTagName (
            IXML_Element * element )
```

Returns the name of the tag as a constant string.

Returns

The name of the **Element**.

Parameters

in	*element*	The **Element** from which to retrieve the name.

7.7.5.34 ixmlElement_hasAttribute()

```
UPNP_EXPORT_SPEC int ixmlElement_hasAttribute (
            IXML_Element * element,
            const char * name )
```

Queries whether the **Element** has an attribute with the given name or a default value.

Returns

`1` if the **Element** has an attribute with this name or has a default value for that attribute, otherwise `0`.

Parameters

in	*element*	The **Element** on which to check for an attribute.
in	*name*	The name of the attribute for which to check.

7.7.5.35 ixmlElement_hasAttributeNS()

```
UPNP_EXPORT_SPEC int ixmlElement_hasAttributeNS (
            IXML_Element * element,
            const char * namespaceURI,
            const char * localName )
```

Queries whether the **Element** has an attribute with the given local name and namespace URI or has a default value for that attribute.

Returns

`1` if the **Element** has an attribute with the given namespace and local name or has a default value for that attribute, otherwise `0`.

Parameters

in	*element*	The **Element** on which to check for the attribute.
in	*namespaceURI*	The namespace URI of the attribute.
in	*localName*	The local name of the attribute.

7.7.5.36 ixmlElement_init()

```
UPNP_EXPORT_SPEC void ixmlElement_init (
            IXML_Element * element )
```

Initializes a **IXML_Element** node.

Parameters

in	*element*	The **Element** to initialize.

Referenced by ixmlNode_cloneElement().

7.7.5.37 ixmlElement_removeAttribute()

```
UPNP_EXPORT_SPEC int ixmlElement_removeAttribute (
            IXML_Element * element,
            const char * name )
```

Removes an attribute value by name. The attribute node is not removed.

Returns

An integer representing one of the following:

- `IXML_SUCCESS`: The operation completed successfully.
- `IXML_INVALID_PARAMETER`: Either **element** or **name** is `NULL`.

Parameters

in	*element*	The **Element** from which to remove the attribute.
in	*name*	The name of the attribute to remove.

7.7.5.38 ixmlElement_removeAttributeNode()

```
UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNode (
            IXML_Element * element,
            IXML_Attr * oldAttr,
            IXML_Attr ** rtAttr )
```

Removes the specified attribute node from an **Element**.

Returns

An integer representing one of the following:

- `IXML_SUCCESS`: The operation completed successfully.
- `IXML_INVALID_PARAMETER`: Either **element** or **oldAttr** is `NULL`.
- `IXML_NOT_FOUND_ERR`: **oldAttr** is not among the list attributes of **element**.

Parameters

in	*element*	The **Element** from which to remove the attribute.
in	*oldAttr*	The attribute to remove from the **Element**.
out	*rtAttr*	A pointer to an attribute in which to place the removed attribute.

References ixmlElement_findAttributeNode().

7.7.5.39 ixmlElement_removeAttributeNS()

```
UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNS (
            IXML_Element * element,
            const char * namespaceURI,
            const char * localName )
```

Removes an attribute using the namespace URI and local name.

The replacing attribute has the same namespace URI and local name, as well as the original prefix.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **element**, **namespaceURI**, or **localName** is NULL.

Parameters

in	*element*	The **Element** from which to remove the the attribute.
in	*namespaceURI*	The namespace URI of the attribute.
in	*localName*	The local name of the attribute.

7.7.5.40 ixmlElement_setAttribute()

```
UPNP_EXPORT_SPEC int ixmlElement_setAttribute (
            IXML_Element * element,
            const char * name,
            const char * value )
```

Adds a new attribute to an **Element**.

If an attribute with the same name already exists in the element, the attribute value will be updated with the new value parameter. Otherwise, a new attribute is inserted into the element.

Returns

An integer representing of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **element**, **name**, or **value** is NULL.
- IXML_INVALID_CHARACTER_ERR: **name** contains an illegal character.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete the operation.

Parameters

in	*element*	The **Element** on which to set the attribute.
in	*name*	The name of the attribute.
in	*value*	The value of the attribute. Note that this is a non-parsed string and any markup must be escaped.

7.7.5.41 ixmlElement_setAttributeNode()

```
UPNP_EXPORT_SPEC int ixmlElement_setAttributeNode (
            IXML_Element * element,
            IXML_Attr * newAttr,
            IXML_Attr ** rtAttr )
```

Adds a new attribute node to an **Element**.

If an attribute already exists with **newAttr** as a name, it will be replaced with the new one and the old one will be returned in **rtAttr**.

Returns

If successfull, the replaced attribute node is returned in rtAttr, otherwise, **NULL** is returned in this pointer. The function return value is an integer representing one of the following:

- `IXML_SUCCESS`: The operation completed successfully.
- `IXML_INVALID_PARAMETER`: Either **element** or **newAttr** is `NULL`.
- `IXML_WRONG_DOCUMENT_ERR`: **newAttr** does not belong to the same one as **element**.
- `IXML_INUSE_ATTRIBUTE_ERR`: **newAttr** is already an attribute of another **Element**.

Parameters

in	*element*	The **Element** in which to add the new attribute.
in	*newAttr*	The new **Attr** to add.
out	*rtAttr*	A pointer to an **Attr** where the old **Attr** will be stored. This will have a `NULL` if no prior node existed.

References ixmlAttr_free().

Referenced by Parser_processAttributeName().

7.7.5.42 ixmlElement_setAttributeNodeNS()

```
UPNP_EXPORT_SPEC int ixmlElement_setAttributeNodeNS (
            IXML_Element * element,
            IXML_Attr * newAttr,
            IXML_Attr ** rcAttr )
```

Adds a new attribute node to the element node specified.

If an attribute with the same local name and namespace URI already exists in the **Element**, the existing attribute node is replaced with **newAttr** and the old returned in **rcAttr**.

Returns

The output parameter rcAttr receives the replaced attribute node if the newAttr attribute replaces an existing attribute with the same local name and namespace, otherwise rcAttr receives **NULL**.

The function return value is an integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **element** or **newAttr** is NULL.
- IXML_WRONG_DOCUMENT_ERR: **newAttr** does not belong to the same document as **element**.
- IXML_INUSE_ATTRIBUTE_ERR: **newAttr** already is an attribute of another **Element**.

Parameters

in	*element*	The **Element** in which to add the attribute node.
in	*newAttr*	The new **Attr** to add.
out	*rcAttr*	A pointer to the replaced **Attr**, if it exists.

7.7.5.43 ixmlElement_setAttributeNS()

```
UPNP_EXPORT_SPEC int ixmlElement_setAttributeNS (
            IXML_Element * element,
            const char * namespaceURI,
            const char * qualifiedName,
            const char * value )
```

Adds a new attribute to an **Element** using the local name and namespace URI.

If another attribute matches the same local name and namespace, the prefix is changed to be the prefix part of the qualifiedName and the value is changed to **value**.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **element**, **namespaceURI**, **qualifiedName**, or **value** is NULL.
- IXML_INVALID_CHARACTER_ERR: **qualifiedName** contains an invalid character.
- IXML_NAMESPACE_ERR: Either the **qualifiedName** or **namespaceURI** is malformed. Refer to the DOM2-Core for possible reasons.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exist to complete the operation.
- IXML_FAILED: The operation could not be completed.

Parameters

in	*element*	The **Element** on which to set the attribute.
in	*namespaceURI*	The namespace URI of the new attribute.
in	*qualifiedName*	The qualified name of the attribute.
in	*value*	The new value for the attribute.

7.7.5.44 ixmlFreeDOMString()

```
UPNP_EXPORT_SPEC void ixmlFreeDOMString (
            char * buf )
```

Frees a **DOMString**.

Parameters

in	*buf*	The **DOMString** to free.

Referenced by free_notify_struct(), freeService(), freeServiceList(), freeServiceTable(), handle_invoke_action(), removeServiceTable(), SampleUtil_PrintEvent(), send_action_response(), TestClass_delete(), TestClass_set_TheDomString(), UpnpDownloadXmlDoc(), UpnpExtraHeaders_delete(), UpnpExtraHeaders_set_resp(), UpnpFileInfo_delete(), UpnpFileInfo_set_ContentType(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpStateVarComplete_delete(), UpnpStateVarComplete_set_CurrentVal(), UpnpStateVarRequest_delete(), and UpnpStateVarRequest_set_CurrentVal().

7.7.5.45 ixmlLoadDocument()

```
UPNP_EXPORT_SPEC IXML_Document * ixmlLoadDocument (
            const char * xmlFile )
```

Parses an XML text file converting it into an IXML DOM representation.

Returns

A **Document** if the file correctly parses or NULL on an error.

Parameters

in	*xmlFile*	The filename of the XML text to convert to a **Document**.

References ixmlLoadDocumentEx().

7.7.5.46 ixmlLoadDocumentEx()

```
UPNP_EXPORT_SPEC int ixmlLoadDocumentEx (
            const char * xmlFile,
            IXML_Document ** doc )
```

Parses an XML text file converting it into an IXML DOM representation.

The **ixmlLoadDocumentEx** API differs from the **ixmlLoadDocument** API in that it returns a an error code representing the actual failure rather than just NULL.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: The **xmlFile** is not a valid pointer.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters

in	*xmlFile*	The filename of the XML text to convert to a **Document**.
out	*doc*	A pointer to the **Document** if file correctly parses or **NULL** on an error.

References Parser_LoadDocument().

Referenced by ixmlLoadDocument().

7.7.5.47 ixmlNamedNodeMap_free()

```
UPNP_EXPORT_SPEC void ixmlNamedNodeMap_free (
            IXML_NamedNodeMap * nnMap )
```

Frees a **NamedNodeMap**.

The **Nodes** inside the map are not freed, just the **NamedNodeMap** object.

Parameters

in	*nnMap*	The **NamedNodeMap** to free.

Referenced by ixmlNode_getAttributes().

7.7.5.48 ixmlNamedNodeMap_getLength()

```
UPNP_EXPORT_SPEC unsigned long ixmlNamedNodeMap_getLength (
            IXML_NamedNodeMap * nnMap )
```

Returns the number of items contained in this **NamedNodeMap**.

Returns

The number of nodes in this map.

Parameters

in	*nnMap*	The **NamedNodeMap** from which to retrieve the size.

Referenced by ixmlNamedNodeMap_item().

7.7.5.49 ixmlNamedNodeMap_getNamedItem()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_getNamedItem (
            IXML_NamedNodeMap * nnMap,
            const char * name )
```

Retrieves a **Node** from the **NamedNodeMap** by name.

Returns

A Node with the specified nodeName, or **NULL** if it does not identify any node in this map.

Parameters

in	*nnMap*	The **NamedNodeMap** to search.
in	*name*	The name of the **Node** to find.

7.7.5.50 ixmlNamedNodeMap_getNamedItemNS()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_getNamedItemNS (
            IXML_NamedNodeMap * nnMap,
            const char ** namespaceURI,
            const char * localName )
```

Retrieves a **Node** from a **NamedNodeMap** specified by namespace URI and local name.

Returns

A pointer to the **Node**, if found, or NULL if it wasn't

Parameters

nnMap	The **NamedNodeMap** from which to remove the **Node**.
namespaceURI	The namespace URI of the **Node** to remove.
localName	The local name of the **Node** to remove.

7.7.5.51 ixmlNamedNodeMap_item()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_item (
            IXML_NamedNodeMap * nnMap,
            unsigned long index )
```

Retrieves the indexth item in the map. If index is greater than or equal to the number of nodes in this map, this returns **NULL**.

Returns

The node at the indexth position in the map, or **NULL** if that is not a valid index.

Parameters

in	*nnMap*	The **NamedNodeMap** from which to remove the **Node**.
in	*index*	The index into the map to remove.

References ixmlNamedNodeMap_getLength().

7.7.5.52 ixmlNamedNodeMap_removeNamedItem()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_removeNamedItem (
             IXML_NamedNodeMap * nnMap,
             const char * name )
```

Removes a **Node** from a **NamedNodeMap** specified by name.

Returns

A pointer to the **Node**, if found, or `NULL` if it wasn't.

Parameters

nnMap	The **NamedNodeMap** from which to remove the item.
name	The name of the item to remove.

7.7.5.53 ixmlNamedNodeMap_removeNamedItemNS()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_removeNamedItemNS (
             IXML_NamedNodeMap * nnMap,
             const char * namespaceURI,
             const char * localName )
```

Removes a **Node** from a **NamedNodeMap** specified by namespace URI and local name.

Returns

A pointer to the **Node**, if found, or `NULL` if it wasn't.

Parameters

nnMap	The **NamedNodeMap** from which to remove the **Node**.
namespaceURI	The namespace URI of the **Node** to remove.
localName	The local name of the **Node** to remove.

7.7.5.54 ixmlNamedNodeMap_setNamedItem()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_setNamedItem (
             IXML_NamedNodeMap * nnMap,
             IXML_Node * arg )
```

Adds a new **Node** to the **NamedNodeMap** using the **Node** name attribute.

Returns

The old **Node** if the new **Node** replaces it or `NULL` if the **Node** was not in the **NamedNodeMap** before.

Parameters

nnMap	The **NamedNodeMap** in which to add the new **Node**.
arg	The new **Node** to add to the **NamedNodeMap**.

7.7.5.55 ixmlNamedNodeMap_setNamedItemNS()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_setNamedItemNS (
              IXML_NamedNodeMap * nnMap,
              IXML_Node * arg )
```

Adds a new **Node** to the **NamedNodeMap** using the **Node** local name and namespace URI attributes.

Returns

The old **Node** if the new **Node** replaces it or NULL if the **Node** was not in the **NamedNodeMap** before.

Parameters

nnMap	The **NamedNodeMap** in which to add the **Node**.
arg	The **Node** to add to the map.

7.7.5.56 ixmlNode_appendChild()

```
UPNP_EXPORT_SPEC int ixmlNode_appendChild (
              IXML_Node * nodeptr,
              IXML_Node * newChild )
```

Appends a child **Node** to the list of children of a **Node**.

If **newChild** is already in the tree, it is removed first.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **nodeptr** or **newChild** is NULL.
- IXML_HIERARCHY_REQUEST_ERR: **newChild** is of a type that cannot be added as a child of **nodeptr** or **newChild** is an ancestor of **nodeptr**.
- IXML_WRONG_DOCUMENT_ERR: **newChild** was created from a different document than **nodeptr**.
- IXML_NO_MODIFICATION_ALLOWED_ERR: **nodeptr** is a read-only **Node**.

Parameters

in	*nodeptr*	The **Node** in which to append the new child.
in	*newChild*	The new child to append.

References ixmlNode_allowChildren(), ixmlNode_isAncestor(), ixmlNode_isParent(), and ixmlNode_removeChild().

Referenced by addToAction(), ixmlNode_insertBefore(), makeAction(), Parser_parseDocument(), Parser_processElementName(), UpnpAddToPropertySet(), and UpnpCreatePropertySet().

7.7.5.57 ixmlNode_cloneNode()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNode_cloneNode (
            IXML_Node * nodeptr,
            int deep )
```

Clones a **Node**.

The new **Node** does not have a parent. The **deep** parameter controls whether the subtree of the **Node** is also cloned.

For details on cloning specific types of **Nodes**, refer to the DOM2-Core recommendation.

Returns

A clone of **nodeptr** or NULL.

Parameters

in	*nodeptr*	The **Node** to clone.
in	*deep*	1 to clone the subtree also or 0 to clone only **nodeptr**.

References ixmlNode_cloneAttrDirect(), and ixmlNode_cloneNodeTree().

Referenced by ixmlDocument_importNode().

7.7.5.58 ixmlNode_free()

```
UPNP_EXPORT_SPEC void ixmlNode_free (
            IXML_Node * nodeptr )
```

Frees a **Node** and all **Nodes** in its subtree.

Parameters

in	*nodeptr*	The **Node** tree to free. Before it is freed, the handler set by **ixmlSetBeforeFree** will be called, the order will be top-down.

References ixmlNode_freeSingleNode().

Referenced by ixmlAttr_free(), ixmlCDATASection_free(), ixmlDocument_free(), ixmlElement_free(), ixmlNode_cloneTextNode(), and ixmlNode_removeChild().

7.7.5.59 ixmlNode_getAttributes()

```
UPNP_EXPORT_SPEC IXML_NamedNodeMap * ixmlNode_getAttributes (
            IXML_Node * nodeptr )
```

Retrieves the attributes of a **Node**, if it is an **Element** node, in a **NamedNodeMap** structure.

Returns

A **NamedNodeMap** of the attributes or NULL.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the attributes.

References ixmlNamedNodeMap_addToNamedNodeMap(), ixmlNamedNodeMap_free(), and ixmlNamedNodeMap_init().

7.7.5.60 ixmlNode_getChildNodes()

```
UPNP_EXPORT_SPEC IXML_NodeList * ixmlNode_getChildNodes (
            IXML_Node * nodeptr )
```

Retrieves the list of children of a **Node** in a **NodeList** structure.

If a **Node** has no children, **ixmlNode_getChildNodes** returns a **NodeList** structure that contains no **Nodes**.

Returns

A **NodeList** of the children of the **Node**.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the children.

References ixmlNodeList_addToNodeList(), ixmlNodeList_free(), and ixmlNodeList_init().

Referenced by printNodes().

7.7.5.61 ixmlNode_getFirstChild()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getFirstChild (
            IXML_Node * nodeptr )
```

Retrieves the first child **Node** of a **Node**.

Returns

A pointer to the first child **Node** or NULL if the **Node** does not have any children.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the first child.

Referenced by addToAction(), AdvertiseAndReply(), check_soap_request(), getElementValue(), getSubElement(), ixmlDocument_setOwnerDocument(), ixmlDomTreetoString(), ixmlNode_getElementsByTagName(), ixmlNode_getElementsByTagNa ixmlNode_getElementsByTagNameNSRecursive(), ixmlNode_getElementsByTagNameRecursive(), ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), makeAction(), SampleUtil_GetElementValue(), SampleUtil_GetFirstDocumentItem(), SampleUtil_GetFirstElementItem(), UpnpAddToPropertySet(), and UpnpCreatePropertySet().

7.7.5.62 ixmlNode_getLastChild()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getLastChild (
             IXML_Node * nodeptr )
```

Retrieves the last child **Node** of a **Node**.

Returns

A pointer to the last child **Node** or NULL if the **Node** does not have any children.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the last child.

7.7.5.63 ixmlNode_getLocalName()

```
UPNP_EXPORT_SPEC const char * ixmlNode_getLocalName (
             IXML_Node * nodeptr )
```

Retrieves the local name of a **Node**, if present.

The local name is the tag name without the namespace prefix. Only **Nodes** of type eELEMENT_NODE or eATTRIBUTE_NODE can have a local name. **Nodes** created through the **Document** interface will only contain a local name if created using **ixmlDocument_createElementNS**.

Returns

A **DOMString** representing the local name of the **Element** or NULL.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the local name.

Referenced by check_soap_request(), ixmlNode_getElementsByTagNameNS(), and ixmlNode_getElementsByTagNameNSRecursive

7.7.5.64 ixmlNode_getNamespaceURI()

```
UPNP_EXPORT_SPEC const char * ixmlNode_getNamespaceURI (
            IXML_Node * nodeptr )
```

Retrieves the namespace URI for a **Node** as a **DOMString**.

Only **Nodes** of type eELEMENT_NODE or eATTRIBUTE_NODE can have a namespace URI. **Nodes** created through the **Document** interface will only contain a namespace if created using **ixmlDocument_createElement↩ NS**.

Returns

A **DOMString** representing the URI of the namespace or NULL.

Parameters

in	*nodeptr*	The **Node** for which to retrieve the namespace.

References DOMString.

Referenced by check_soap_request(), ixmlNode_getElementsByTagNameNS(), and ixmlNode_getElementsByTagNameNSRecursive

7.7.5.65 ixmlNode_getNextSibling()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getNextSibling (
            IXML_Node * nodeptr )
```

Retrieves the sibling **Node** immediately following this **Node**.

Returns

A pointer to the next sibling **Node** or NULL if no such **Node** exists.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the next sibling.

Referenced by AdvertiseAndReply(), getSubElement(), ixmlDocument_setOwnerDocument(), ixmlNode_getElementsByTagNameNSI
ixmlNode_getElementsByTagNameRecursive(), and ixmlPrintDomTreeRecursive().

7.7.5.66 ixmlNode_getNodeName()

```
UPNP_EXPORT_SPEC const char * ixmlNode_getNodeName (
            IXML_Node * nodeptr )
```

Returns the name of the **Node**, depending on what type of **Node** it is, in a read-only string.

Refer to the table in the DOM2-Core for a description of the node names for various interfaces.

Returns

A constant **DOMString** of the node name.

Parameters

in	*nodeptr*	Pointer to the node to retrieve the name.

Referenced by AdvertiseAndReply(), getSubElement(), ixmlDomTreetoString(), ixmlNode_getElementsByTagName(), ixmlNode_getElementsByTagNameRecursive(), ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), and printNodes().

7.7.5.67 ixmlNode_getNodeType()

```
UPNP_EXPORT_SPEC unsigned short ixmlNode_getNodeType (
            IXML_Node * nodeptr )
```

Retrieves the type of a **Node**. Note that not all possible return values are actually implemented.

Returns

An enum IXML_NODE_TYPE representing the type of the **Node**.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the type.

Referenced by check_soap_request(), getElementValue(), getSubElement(), ixmlDocument_importNode(), ixmlDomTreetoString(), ixmlNode_getElementsByTagName(), ixmlNode_getElementsByTagNameNS(), ixmlNode_getElementsByTag ixmlNode_getElementsByTagNameRecursive(), ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), printNodes(), and SampleUtil_GetElementValue().

7.7.5.68 ixmlNode_getNodeValue()

```
UPNP_EXPORT_SPEC const char * ixmlNode_getNodeValue (
            IXML_Node * nodeptr )
```

Returns the value of the **Node** as a string.

Note that this string is not a copy and modifying it will modify the value of the **Node**.

Returns

A **DOMString** of the **Node** value.

Parameters

in	*nodeptr*	Pointer to the **Node** to retrieve the value.

Referenced by AdvertiseAndReply(), getElementValue(), handle_query_variable(), ixmlDomTreetoString(),

ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), printNodes(), SampleUtil_GetElementValue(), SampleUtil_GetFirstDocumentItem() and SampleUtil_GetFirstElementItem().

7.7.5.69 ixmlNode_getOwnerDocument()

```
UPNP_EXPORT_SPEC IXML_Document * ixmlNode_getOwnerDocument (
            IXML_Node * nodeptr )
```

Retrieves the document object associated with this **Node**.

This owner document **Node** allows other **Nodes** to be created in the context of this document. Note that **Document** nodes do not have an owner document.

Returns

A pointer to the owning **Document** or NULL, if the **Node** does not have an owner.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the owner document.

7.7.5.70 ixmlNode_getParentNode()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getParentNode (
            IXML_Node * nodeptr )
```

Retrieves the parent **Node** for a **Node**.

Returns

A pointer to the parent **Node** or NULL if the **Node** has no parent.

Parameters

in	*nodeptr*	The **Node** from which to retrieve the parent.

7.7.5.71 ixmlNode_getPrefix()

```
UPNP_EXPORT_SPEC const char * ixmlNode_getPrefix (
            IXML_Node * nodeptr )
```

Retrieves the namespace prefix, if present.

The prefix is the name used as an alias for the namespace URI for this element. Only **Nodes** of type eELEMENT_NODE or eATTRIBUTE_NODE can have a prefix. **Nodes** created through the **Document** interface will only contain a prefix if created using **ixmlDocument_createElementNS**.

Returns

A **DOMString** representing the namespace prefix or NULL.

Parameters

nodeptr	The **Node** from which to retrieve the prefix.

References DOMString.

7.7.5.72 ixmlNode_getPreviousSibling()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getPreviousSibling (
            IXML_Node * nodeptr )
```

Retrieves the sibling **Node** immediately preceding this **Node**.

Returns

A pointer to the previous sibling **Node** or `NULL` if no such **Node** exists.

Parameters

`in`	*nodeptr*	The **Node** for which to retrieve the previous sibling.

7.7.5.73 ixmlNode_hasAttributes()

```
UPNP_EXPORT_SPEC int ixmlNode_hasAttributes (
            IXML_Node * nodeptr )
```

Queries whether this **Node** has attributes.

Note that only **Element** nodes have attributes.

Returns

`1` if the **Node** has attributes otherwise `0`.

Parameters

`in`	*nodeptr*	The **Node** to query for attributes.

7.7.5.74 ixmlNode_hasChildNodes()

```
UPNP_EXPORT_SPEC int ixmlNode_hasChildNodes (
            IXML_Node * nodeptr )
```

Queries whether or not a **Node** has children.

Returns

`1` if the **Node** has one or more children otherwise `0`.

Parameters

in	*nodeptr*	The **Node** to query for children.

7.7.5.75 ixmlNode_insertBefore()

```
UPNP_EXPORT_SPEC int ixmlNode_insertBefore (
            IXML_Node * nodeptr,
            IXML_Node * newChild,
            IXML_Node * refChild )
```

Inserts a new child **Node** before the existing child **Node**.

refChild can be NULL, which inserts **newChild** at the end of the list of children. Note that the **Node** (or **Nodes**) in **newChild** must already be owned by the owner document (or have no owner at all) of **nodeptr** for insertion. If not, the **Node** (or **Nodes**) must be imported into the document using **ixmlDocument_importNode**. If **newChild** is already in the tree, it is removed first.

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **nodeptr** or **newChild** is NULL.
- IXML_HIERARCHY_REQUEST_ERR: The type of the **Node** does not allow children of the type of **newChild**.
- IXML_WRONG_DOCUMENT_ERR: **newChild** has an owner document that does not match the owner of **nodeptr**.
- IXML_NO_MODIFICATION_ALLOWED_ERR: **nodeptr** is read-only or the parent of the **Node** being inserted is read-only.
- IXML_NOT_FOUND_ERR: **refChild** is not a child of **nodeptr**.

Parameters

in	*nodeptr*	The parent of the **Node** before which to insert the new child.
in	*newChild*	The **Node** to insert into the tree.
in	*refChild*	The reference child where the new **Node** should be inserted. The new **Node** will appear directly before the reference child.

References ixmlNode_allowChildren(), ixmlNode_appendChild(), ixmlNode_isAncestor(), ixmlNode_isParent(), and ixmlNode_removeChild().

Referenced by ixmlNode_replaceChild().

7.7.5.76 ixmlNode_removeChild()

```
UPNP_EXPORT_SPEC int ixmlNode_removeChild (
            IXML_Node * nodeptr,
```

```
            IXML_Node * oldChild,
            IXML_Node ** returnNode )
```

Removes a child from the list of children of a **Node**.

returnNode will contain the **oldChild Node**, appropriately removed from the tree (i.e. it will no longer have an owner document).

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **nodeptr** or **oldChild** is NULL.
- IXML_NO_MODIFICATION_ALLOWED_ERR: **nodeptr** or its parent is read-only.
- IXML_NOT_FOUND_ERR: **oldChild** is not among the children of **nodeptr**.

Parameters

in	*nodeptr*	The parent of the child to remove.
in	*oldChild*	The child **Node** to remove.
out	*returnNode*	Pointer to a **Node** to place the removed **oldChild Node**.

References ixmlNode_free(), and ixmlNode_isParent().

Referenced by ixmlNode_appendChild(), ixmlNode_insertBefore(), and ixmlNode_replaceChild().

7.7.5.77 ixmlNode_replaceChild()

```
UPNP_EXPORT_SPEC int ixmlNode_replaceChild (
            IXML_Node * nodeptr,
            IXML_Node * newChild,
            IXML_Node * oldChild,
            IXML_Node ** returnNode )
```

Replaces an existing child **Node** with a new child **Node** in the list of children of a **Node**.

If **newChild** is already in the tree, it will first be removed. **returnNode** will contain the **oldChild Node**, appropriately removed from the tree (i.e. it will no longer have an owner document).

Returns

An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMTER: Either **nodeptr**, **newChild**, or **oldChild** is NULL.
- IXML_HIERARCHY_REQUEST_ERR: The **newChild** is not a type of **Node** that can be inserted into this tree or **newChild** is an ancestor of **nodePtr**.
- IXML_WRONG_DOCUMENT_ERR: **newChild** was created from a different document than **nodeptr**.
- IXML_NO_MODIFICATION_ALLOWED_ERR: **nodeptr** or its parent is read-only.
- IXML_NOT_FOUND_ERR: **oldChild** is not a child of **nodeptr**.

Parameters

in	*nodeptr*	The parent of the **Node** which contains the child to replace.
in	*newChild*	The child with which to replace **oldChild**.
in	*oldChild*	The child to replace with **newChild**.
out	*returnNode*	Pointer to a **Node** to place the removed **oldChild Node**.

References ixmlNode_allowChildren(), ixmlNode_insertBefore(), ixmlNode_isAncestor(), ixmlNode_isParent(), and ixmlNode_removeChild().

7.7.5.78 ixmlNode_setNodeValue()

```
UPNP_EXPORT_SPEC int ixmlNode_setNodeValue (
            IXML_Node * nodeptr,
            const char * newNodeValue )
```

Assigns a new value to a **Node**.

The **newNodeValue** string is duplicated and stored in the **Node** so that the original does not have to persist past this call.

Returns

An integer representing one of the following:

- `IXML_SUCCESS`: The operation completed successfully.
- `IXML_INVALID_PARAMETER`: The **Node** * is not a valid pointer.
- `IXML_INSUFFICIENT_MEMORY`: Not enough free memory exists to complete this operation.

Parameters

in	*nodeptr*	The **Node** to which to assign a new value.
in	*newNodeValue*	The new value of the **Node**.

Referenced by ixmlNode_cloneAttr(), ixmlNode_cloneCDATASect(), ixmlNode_cloneElement(), ixmlNode_cloneTextNode(), and ixmlNode_setNodeProperties().

7.7.5.79 ixmlNodeList_free()

```
UPNP_EXPORT_SPEC void ixmlNodeList_free (
            IXML_NodeList * nList )
```

Frees a **NodeList** object.

Since the underlying **Nodes** are references, they are not freed using this operation. This only frees the **NodeList** object.

Parameters

in	*nList*	The **NodeList** to free.

Referenced by AdvertiseAndReply(), ixmlNode_getChildNodes(), removeServiceTable(), SampleUtil_FindAndParseService(), SampleUtil_GetFirstDocumentItem(), SampleUtil_GetFirstElementItem(), SampleUtil_GetFirstServiceList(), TvStateUpdate(), and UpnpUnRegisterRootDeviceLowPower().

7.7.5.80 ixmlNodeList_item()

```
UPNP_EXPORT_SPEC IXML_Node * ixmlNodeList_item (
             IXML_NodeList * nList,
             unsigned long index )
```

Retrieves a **Node** from a **NodeList** specified by a numerical index.

Returns

A pointer to a **Node** or NULL if there was an error.

Parameters

in	*nList*	The **NodeList** from which to retrieve the **Node**.
in	*index*	The index into the **NodeList** to retrieve.

References ixmlNodeList_length().

Referenced by AdvertiseAndReply(), printNodes(), SampleUtil_FindAndParseService(), SampleUtil_GetFirstDocumentItem(), SampleUtil_GetFirstElementItem(), SampleUtil_GetFirstServiceList(), and TvStateUpdate().

7.7.5.81 ixmlNodeList_length()

```
UPNP_EXPORT_SPEC unsigned long ixmlNodeList_length (
             IXML_NodeList * nList )
```

Returns the number of **Nodes** in a **NodeList**.

Returns

The number of **Nodes** in the **NodeList**.

Parameters

in	*nList*	The **NodeList** for which to retrieve the number of **Nodes**.

Referenced by ixmlNodeList_item(), removeServiceTable(), SampleUtil_FindAndParseService(), SampleUtil_GetFirstServiceList(), and TvStateUpdate().

7.7.5.82 ixmlNodetoString()

```
UPNP_EXPORT_SPEC char * ixmlNodetoString (
             IXML_Node * doc )
```

Renders a **Node** and all sub-elements into an XML text representation. The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.

Note that this function can be used for any **Node-derived** interface. The difference between **ixmlNodetoString** and **ixmlDocumenttoString** is **ixmlNodetoString** does not include the XML prolog, it only produces XML elements.

Returns

A **DOMString** with the XML text representation of the DOM tree or `NULL` on an error.

Parameters

`in`	*doc*	The root of the **Node** tree to render to XML text.

References ixml_membuf_init(), and ixmlDomTreetoString().

7.7.5.83 ixmlParseBuffer()

```
UPNP_EXPORT_SPEC IXML_Document * ixmlParseBuffer (
             const char * buffer )
```

Parses an XML text buffer converting it into an IXML DOM representation.

Returns

A **Document** if the buffer correctly parses or `NULL` on an error.

Parameters

`in`	*buffer*	The buffer that contains the XML text to convert to a **Document**.

References ixmlParseBufferEx().

7.7.5.84 ixmlParseBufferEx()

```
UPNP_EXPORT_SPEC int ixmlParseBufferEx (
             const char * buffer,
             IXML_Document ** doc )
```

Parses an XML text buffer converting it into an IXML DOM representation.

The **ixmlParseBufferEx** API differs from the **ixmlParseBuffer** API in that it returns an error code representing the actual failure rather than just `NULL`.

Returns

An integer representing one of the following:

- `IXML_SUCCESS`: The operation completed successfully.
- `IXML_INVALID_PARAMETER`: The **buffer** is not a valid pointer.
- `IXML_INSUFFICIENT_MEMORY`: Not enough free memory exists to complete this operation.

Parameters

in	*buffer*	The buffer that contains the XML text to convert to a **Document**.
out	*doc*	A point to store the **Document** if file correctly parses or **NULL** on an error.

References Parser_LoadDocument().

Referenced by addToAction(), gena_process_notification_event(), handle_invoke_action(), ixmlParseBuffer(), makeAction(), soap_device_callback(), UpnpAddToPropertySet(), UpnpCreatePropertySet(), UpnpDownloadXmlDoc(), UpnpSendActionAsync(), and UpnpSendActionExAsync().

7.7.5.85 ixmlPrintDocument()

```
UPNP_EXPORT_SPEC char * ixmlPrintDocument (
            IXML_Document * doc )
```

Renders a **Node** and all sub-elements into an XML document representation.

The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.

Note that this function can be used for any **Node-derived** interface. The difference between **ixmlPrintDocument** and **ixmlPrintNode** is **ixmlPrintDocument** includes the XML prolog while **ixmlPrintNode** only produces XML elements. An XML document is not well formed unless it includes the prolog and at least one element.

This function introduces lots of white space to print the **DOMString** in readable format.

Returns

A **DOMString** with the XML document representation of the DOM tree or NULL on an error.

Parameters

in	*doc*	The document node to print.

References ixml_membuf_append_str(), ixml_membuf_init(), and ixmlPrintDomTree().

7.7.5.86 ixmlPrintNode()

```
UPNP_EXPORT_SPEC char * ixmlPrintNode (
            IXML_Node * doc )
```

Renders a **Node** and all sub-elements into an XML text representation.

The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.

Note that this function can be used for any **Node-derived** interface. A similar **ixmlPrintDocument** function is defined to avoid casting when printing whole documents. This function introduces lots of white space to print the **DOMString** in readable format.

Returns

A **DOMString** with the XML text representation of the DOM tree or NULL on an error.

Parameters

in	*doc*	The root of the **Node** tree to render to XML text.

References ixml_membuf_init(), and ixmlPrintDomTree().

Referenced by genaInitNotifyExt(), genaNotifyAllExt(), handle_invoke_action(), SampleUtil_PrintEvent(), send_action_response(), UpnpDownloadXmlDoc(), UpnpSendActionAsync(), and UpnpSendActionExAsync().

7.7.5.87 ixmlRelaxParser()

```
UPNP_EXPORT_SPEC void ixmlRelaxParser (
            char errorChar )
```

Makes the XML parser more tolerant to malformed text.

Parameters

in	*errorChar*	If **errorChar** is 0 (default), the parser is strict about XML encoding : invalid UTF-8 sequences or "&" entities are rejected, and the parsing aborts.

If **errorChar** is not 0, the parser is relaxed: invalid UTF-8 characters are replaced by the **errorChar**, and invalid "&" entities are left untranslated. The parsing is then allowed to continue.

References Parser_setErrorChar().

Chapter 8

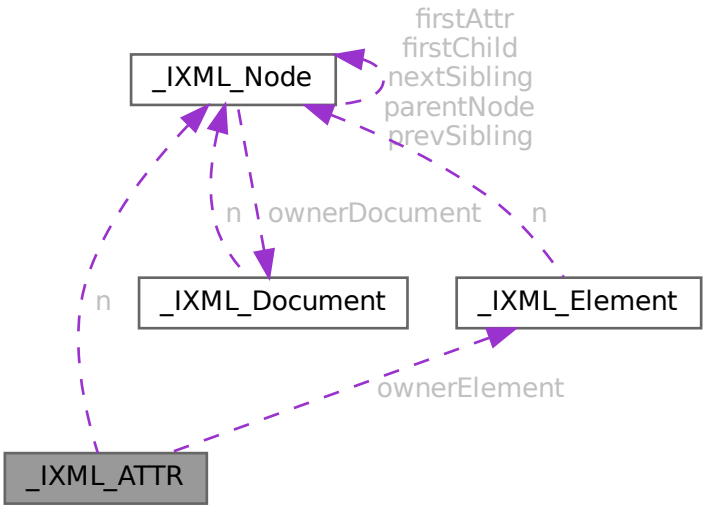
Data Structure Documentation

8.1 _IXML_ATTR Struct Reference

Data structure representing an Attribute node.

`#include <ixml.h>`

Collaboration diagram for _IXML_ATTR:



Data Fields

- IXML_Node **n**
- int **specified**
- IXML_Element * **ownerElement**

8.1.1 Detailed Description

Data structure representing an Attribute node.

The documentation for this struct was generated from the following file:

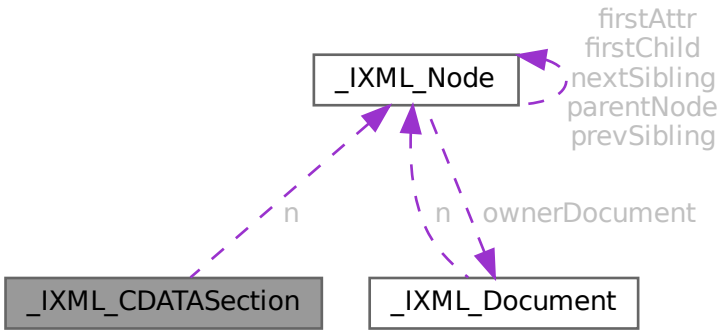
- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.2 _IXML_CDATASection Struct Reference

Data structure representing a CDATA section node.

`#include <ixml.h>`

Collaboration diagram for _IXML_CDATASection:



Data Fields

- IXML_Node **n**

8.2.1 Detailed Description

Data structure representing a CDATA section node.

The documentation for this struct was generated from the following file:

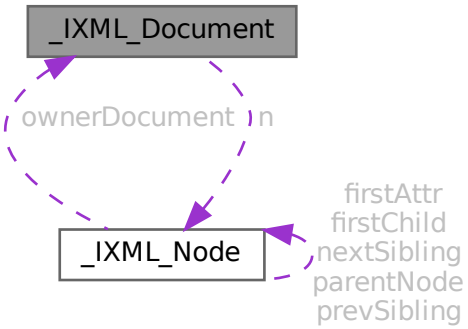
- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.3 _IXML_Document Struct Reference

Data structure representing the DOM Document.

`#include <ixml.h>`

Collaboration diagram for _IXML_Document:



Data Fields

- IXML_Node **n**

8.3.1 Detailed Description

Data structure representing the DOM Document.

The documentation for this struct was generated from the following file:

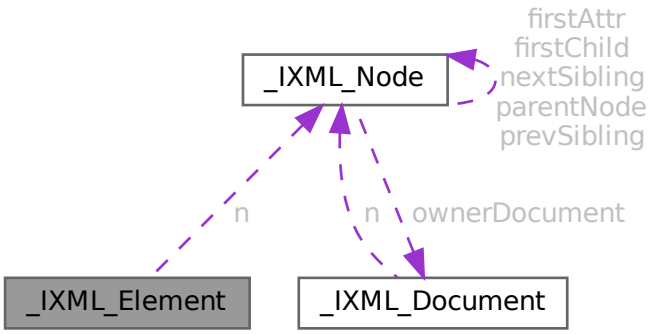
- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.4 _IXML_Element Struct Reference

Data structure representing an Element node.

`#include <ixml.h>`

Collaboration diagram for _IXML_Element:



Data Fields

- IXML_Node **n**
- char * **tagName**

8.4.1 Detailed Description

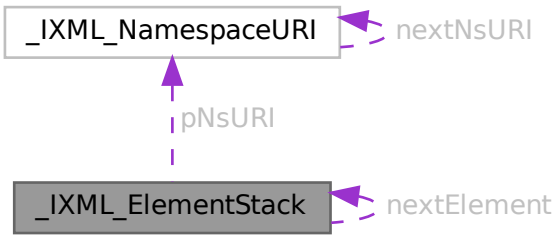
Data structure representing an Element node.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.5 _IXML_ElementStack Struct Reference

Collaboration diagram for _IXML_ElementStack:



Data Fields

- char * **element**
- char * **prefix**
- char * **namespaceUri**
- IXML_NamespaceURI * **pNsURI**
- struct _IXML_ElementStack * **nextElement**

The documentation for this struct was generated from the following file:

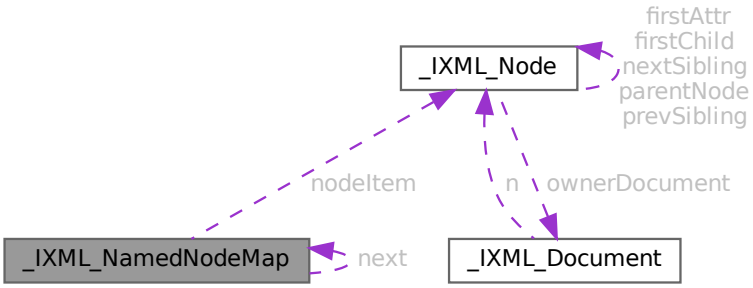
- /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlparser.h

8.6 _IXML_NamedNodeMap Struct Reference

Data structure representing a list of named nodes.

`#include <ixml.h>`

Collaboration diagram for _IXML_NamedNodeMap:



Data Fields

- IXML_Node * **nodeItem**
- struct _IXML_NamedNodeMap * **next**

8.6.1 Detailed Description

Data structure representing a list of named nodes.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.7 _IXML_NamespaceURI Struct Reference

Collaboration diagram for _IXML_NamespaceURI:

Data Fields

- char * **nsURI**
- char * **prefix**
- struct _IXML_NamespaceURI * **nextNsURI**

The documentation for this struct was generated from the following file:

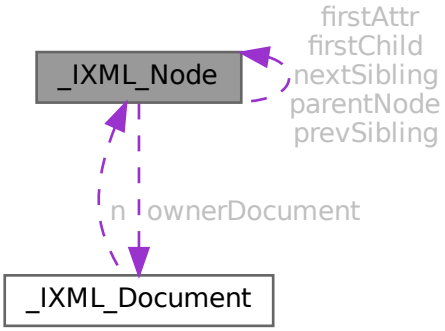
- /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlparser.h

8.8 _IXML_Node Struct Reference

Data structure common to all types of nodes.

`#include <ixml.h>`

Collaboration diagram for _IXML_Node:



Data Fields

- char * **nodeName**
- char * **nodeValue**
- IXML_NODE_TYPE **nodeType**
- char * **namespaceURI**
- char * **prefix**
- char * **localName**
- int **readOnly**
- Nodeptr **parentNode**
- Nodeptr **firstChild**
- Nodeptr **prevSibling**
- Nodeptr **nextSibling**
- Nodeptr **firstAttr**
- Docptr **ownerDocument**

8.8.1 Detailed Description

Data structure common to all types of nodes.

The documentation for this struct was generated from the following file:

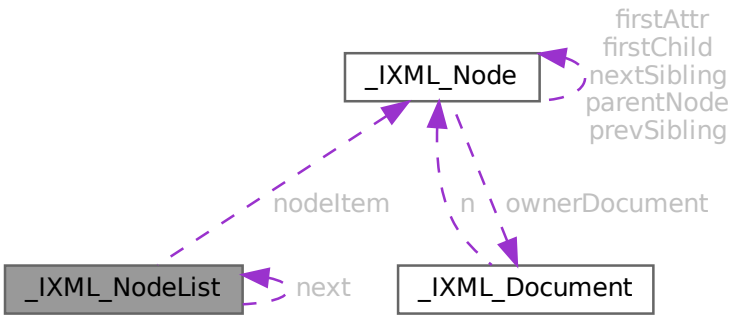
- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.9 _IXML_NodeList Struct Reference

Data structure representing a list of nodes.

`#include <ixml.h>`

Collaboration diagram for _IXML_NodeList:



Data Fields

- IXML_Node * **nodeItem**
- struct _IXML_NodeList * **next**

8.9.1 Detailed Description

Data structure representing a list of nodes.

The documentation for this struct was generated from the following file:

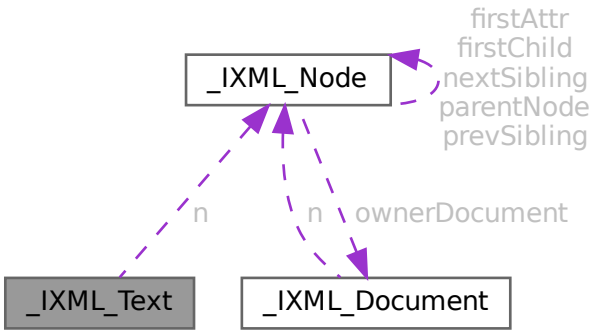
- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.10 _IXML_Text Struct Reference

Data structure representing a Text node.

`#include <ixml.h>`

Collaboration diagram for _IXML_Text:



Data Fields

- IXML_Node **n**

8.10.1 Detailed Description

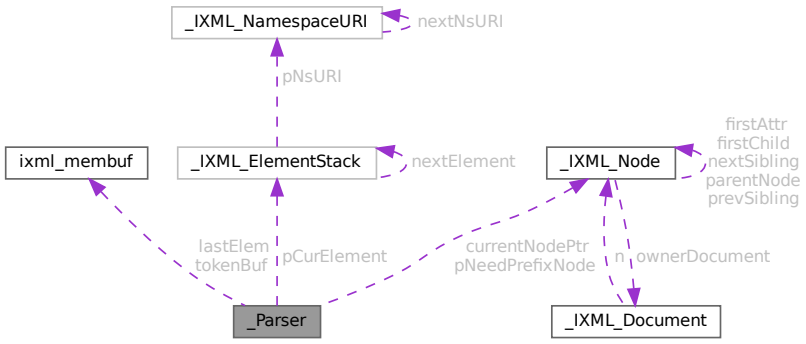
Data structure representing a Text node.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h

8.11 _Parser Struct Reference

Collaboration diagram for _Parser:



Data Fields

- char * dataBuffer
- char * curPtr
- char * savePtr
- ixml_membuf **lastElem**
- ixml_membuf **tokenBuf**
- IXML_Node * **pNeedPrefixNode**
- IXML_ElementStack * **pCurElement**
- IXML_Node * **currentNodePtr**
- PARSER_STATE **state**
- int **bHasTopLevel**

8.11.1 Field Documentation

8.11.1.1 curPtr

```
char* _Parser::curPtr
```

Pointer to the token parsed.

Referenced by Parser_getNextNode(), Parser_getNextToken(), Parser_LoadDocument(), Parser_processAttribute(), Parser_processContent(), Parser_processElementName(), Parser_processSTag(), Parser_skipBom(), Parser_skipMisc(), Parser_skipProlog(), Parser_skipWhiteSpaces(), and Parser_skipXMLDecl().

8.11.1.2 dataBuffer

```
char* _Parser::dataBuffer
```

Data buffer.

Referenced by Parser_free(), Parser_LoadDocument(), and Parser_readFileOrBuffer().

8.11.1.3 savePtr

```
char* _Parser::savePtr
```

Saves for backup.

Referenced by Parser_getNextToken(), Parser_processAttribute(), Parser_processContent(), Parser_processElementName(), and Parser_processSTag().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlparser.h

8.12 _uuid_upnp Struct Reference

```
#include <uuid.h>
```

Data Fields

- uint32_t time_low
- uint16_t time_mid
- uint16_t time_hi_and_version
- uint8_t clock_seq_hi_and_reserved
- uint8_t clock_seq_low
- uint8_t node [6]

8.12.1 Detailed Description

8.12.2 Field Documentation

8.12.2.1 clock_seq_hi_and_reserved

```
uint8_t _uuid_upnp::clock_seq_hi_and_reserved
```

Referenced by format_uuid_v1(), format_uuid_v3(), upnp_uuid_unpack(), and uuid_compare().

8.12.2.2 clock_seq_low

```
uint8_t _uuid_upnp::clock_seq_low
```

Referenced by format_uuid_v1(), upnp_uuid_unpack(), and uuid_compare().

8.12.2.3 node

```
uint8_t _uuid_upnp::node[6]
```

Referenced by format_uuid_v1(), upnp_uuid_unpack(), and uuid_compare().

8.12.2.4 time_hi_and_version

```
uint16_t _uuid_upnp::time_hi_and_version
```

Referenced by format_uuid_v1(), format_uuid_v3(), upnp_uuid_unpack(), uuid_compare(), and uuid_create_from_name().

8.12.2.5 time_low

```
uint32_t _uuid_upnp::time_low
```

Referenced by format_uuid_v1(), format_uuid_v3(), upnp_uuid_unpack(), uuid_compare(), and uuid_create_from_name().

8.12.2.6 time_mid

```
uint16_t _uuid_upnp::time_mid
```

Referenced by format_uuid_v1(), format_uuid_v3(), upnp_uuid_unpack(), uuid_compare(), and uuid_create_from_name().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uuid.h

8.13 char_info Struct Reference

Data Fields

- unsigned short **l**
- unsigned short **h**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlparser.c

8.14 cmdloop_commands Struct Reference

Data Fields

- const char * **str**
- int **cmdnum**
- int **numargs**
- const char * **args**

8.14.1 Detailed Description

Data structure for parsing commands from the command line.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.c

8.15 document_type_t Struct Reference

Data Fields

- const char * file_ext
- const char * content_type
- const char * content_subtype

8.15.1 Field Documentation

8.15.1.1 content_subtype

```
const char* document_type_t::content_subtype
```

Referenced by media_list_init(), and search_extension().

8.15.1.2 content_type

```
const char* document_type_t::content_type
```

Referenced by media_list_init(), and search_extension().

8.15.1.3 file_ext

```
const char* document_type_t::file_ext
```

Referenced by media_list_init(), and search_extension().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/webserver.c

8.16 ErrorString Struct Reference

Structure to maintain a error code and string associated with the error code.

Data Fields

- int rc
- const char * rcError

8.16.1 Detailed Description

Structure to maintain a error code and string associated with the error code.

8.16.2 Field Documentation

8.16.2.1 rc

```
int ErrorString::rc
```

Error code.

Referenced by addToAction(), makeAction(), UpnpAddToPropertySet(), and UpnpGetErrorMessage().

8.16.2.2 rcError

```
const char* ErrorString::rcError
```

Error description.

Referenced by UpnpGetErrorMessage().

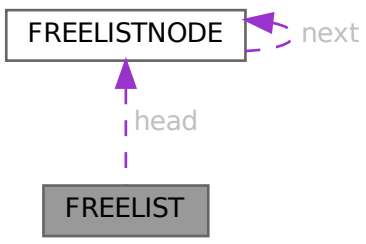
The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnptools.c

8.17 FREELIST Struct Reference

```
#include <FreeList.h>
```

Collaboration diagram for FREELIST:



Data Fields

- FreeListNode * **head**
- size_t **element_size**
- int **maxFreeListLength**
- int **freeListLength**

8.17.1 Detailed Description

Stores head and size of free list, as well as mutex for protection.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/FreeList.h

8.18 FREELISTNODE Struct Reference

`#include <FreeList.h>`

Collaboration diagram for FREELISTNODE:

Data Fields

- struct FREELISTNODE * **next**

8.18.1 Detailed Description

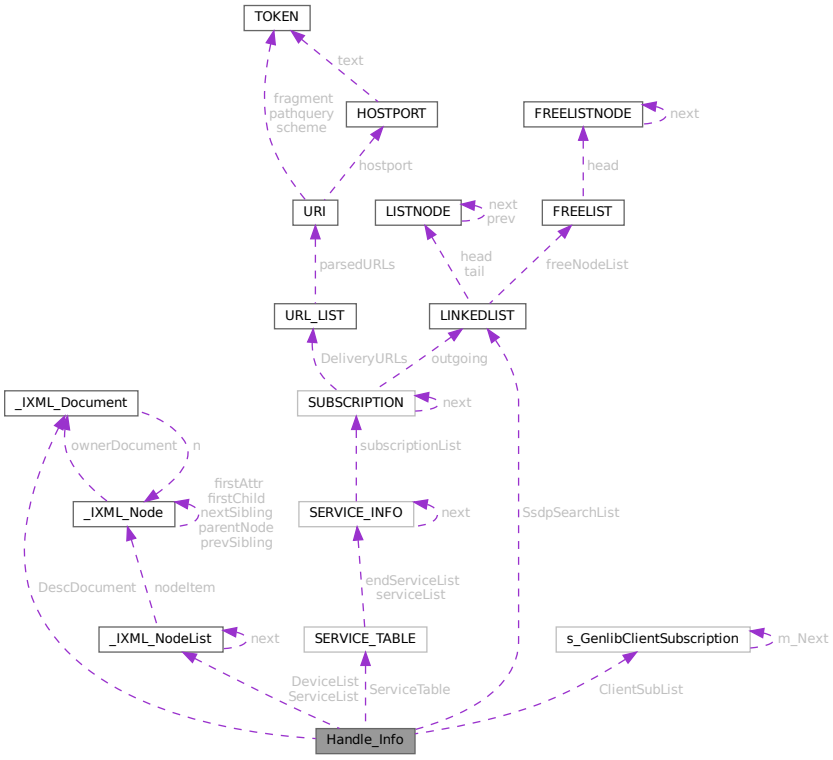
Free list node. points to next free item. Memory for node is borrowed from allocated items.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/FreeList.h

8.19 Handle_Info Struct Reference

Collaboration diagram for Handle_Info:



Data Fields

- Upnp_Handle_Type HType
- Upnp_FunPtr Callback
- char * Cookie
- int aliasInstalled
- char DescURL [LINE_SIZE]
- char LowerDescURL [LINE_SIZE]
- char DescXML [LINE_SIZE]
- int **MaxAge**
- int **PowerState**
- int **SleepPeriod**
- int **RegistrationState**
- IXML_Document * DescDocument
- IXML_NodeList * DeviceList
- IXML_NodeList * ServiceList
- service_table ServiceTable
- int MaxSubscriptions
- int MaxSubscriptionTimeOut
- int DeviceAf
- GenlibClientSubscription * ClientSubList
- LinkedList SsdpSearchList

8.19.1 Field Documentation

8.19.1.1 aliasInstalled

```
int Handle_Info::aliasInstalled
```

0 = not installed; otherwise installed.

Referenced by UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpUnRegisterRootDeviceLowPower().

8.19.1.2 Callback

```
Upnp_FunPtr Handle_Info::Callback
```

Callback function pointer.

Referenced by gena_process_notification_event(), gena_process_subscription_request(), GenaAutoRenewSubscription(), get_dev_service(), ssdp_handle_ctrlpt_msg(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

8.19.1.3 ClientSubList

```
GenlibClientSubscription* Handle_Info::ClientSubList
```

Client subscription list.

Referenced by gena_process_notification_event(), genaRenewSubscription(), genaSubscribe(), genaUnregisterClient(), genaUnSubscribe(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

8.19.1.4 Cookie

```
char* Handle_Info::Cookie
```

Referenced by gena_process_notification_event(), gena_process_subscription_request(), GenaAutoRenewSubscription(), get_dev_service(), ssdp_handle_ctrlpt_msg(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice3(), and UpnpRegisterRootDevice4().

8.19.1.5 DescDocument

```
IXML_Document* Handle_Info::DescDocument
```

Description parsed in terms of DOM document.

Referenced by UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpUnRegisterRootDeviceLowPower().

8.19.1.6 DescURL

```
char Handle_Info::DescURL[LINE_SIZE]
```

URL for the use of SSDP.

Referenced by AdvertiseAndReply(), PrintHandleInfo(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

8.19.1.7 DescXML

```
char Handle_Info::DescXML[LINE_SIZE]
```

XML file path for device description.

8.19.1.8 DeviceAf

```
int Handle_Info::DeviceAf
```

Address family: AF_INET or AF_INET6.

Referenced by AdvertiseAndReply(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpUnRegisterRootDeviceLowPower().

8.19.1.9 DeviceList

```
IXML_NodeList* Handle_Info::DeviceList
```

List of devices in the description document.

Referenced by AdvertiseAndReply(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpUnRegisterRootDeviceLowPower().

8.19.1.10 HType

```
Upnp_Handle_Type Handle_Info::HType
```

Referenced by GetHandleInfo(), PrintHandleInfo(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

8.19.1.11 LowerDescURL

```
char Handle_Info::LowerDescURL[LINE_SIZE]
```

URL for the use of SSDP when answering to legacy CPs (CP searching for a v1 when the device is v2).

Referenced by AdvertiseAndReply(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

8.19.1.12 MaxSubscriptions

```
int Handle_Info::MaxSubscriptions
```

Referenced by gena_process_subscription_renewal_request(), gena_process_subscription_request(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpSetMaxSubscriptions().

8.19.1.13 MaxSubscriptionTimeOut

```
int Handle_Info::MaxSubscriptionTimeOut
```

Referenced by gena_process_subscription_renewal_request(), gena_process_subscription_request(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpSetMaxSubscriptionTimeOut().

8.19.1.14 ServiceList

```
IXML_NodeList* Handle_Info::ServiceList
```

List of services in the description document.

Referenced by UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), and UpnpUnRegisterRootDeviceLowPower().

8.19.1.15 ServiceTable

```
service_table Handle_Info::ServiceTable
```

Table holding subscriptions and URL information.

Referenced by genaNotifyThread(), genaUnregisterDevice(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), and UpnpRegisterRootDevice4().

8.19.1.16 SsdpSearchList

```
LinkedList Handle_Info::SsdpSearchList
```

Active SSDP searches.

Referenced by SearchByTarget(), ssdp_handle_ctrlpt_msg(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice4(), UpnpUnRegisterClient(), and UpnpUnRegisterRootDeviceLowPower().

The documentation for this struct was generated from the following file:

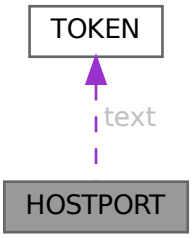
- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnpapi.h

8.20 HOSTPORT Struct Reference

Represents a host port: e.g. "127.127.0.1:80" text is a token pointing to the full string representation.

`#include <uri.h>`

Collaboration diagram for HOSTPORT:



Data Fields

- token text
- struct sockaddr_storage **IPaddress**

8.20.1 Detailed Description

Represents a host port: e.g. "127.127.0.1:80" text is a token pointing to the full string representation.

8.20.2 Field Documentation

8.20.2.1 text

`token HOSTPORT::text`

Full host port.

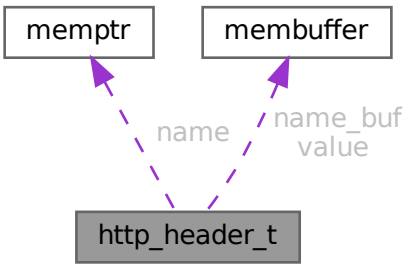
Referenced by copy_URL_list(), create_url_list(), http_FixUrl(), http_MakeMessage(), notify_send_and_recv(), parse_hostport(), and resolve_rel_url().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h

8.21 http_header_t Struct Reference

Collaboration diagram for http_header_t:



Data Fields

- memptr name
- int name_id
- membuffer value
- membuffer **name_buf**

8.21.1 Field Documentation

8.21.1.1 name

```
memptr http_header_t::name
```

header name as a string.

Referenced by Check_Connect_And_Wait_Connection(), CheckOtherHTTPHeaders(), ExtraHTTPHeaders(), and print_http_headers().

8.21.1.2 name_id

```
int http_header_t::name_id
```

header name id (for a selective group of headers only).

8.21.1.3 value

```
membuffer http_header_t::value
```

raw-value; could be multi-lined; min-length = 0.

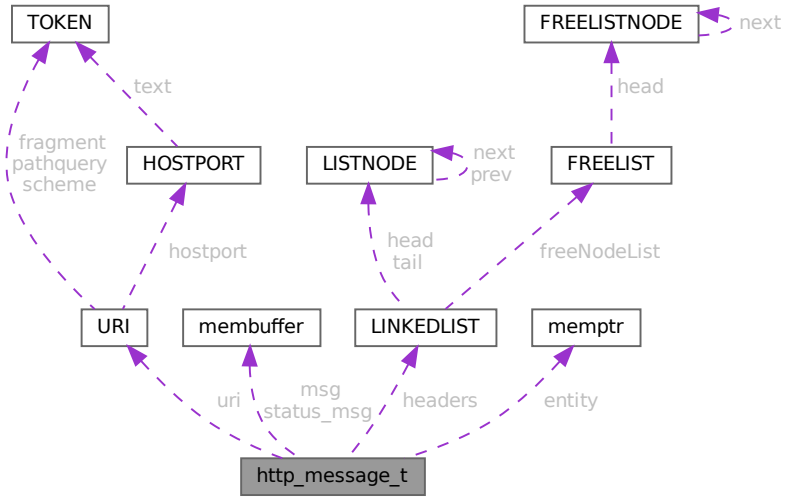
Referenced by Check_Connect_And_Wait_Connection(), CheckOtherHTTPHeaders(), ExtraHTTPHeaders(), get_mpost_acton_hdrval(), and print_http_headers().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h

8.22 http_message_t Struct Reference

Collaboration diagram for http_message_t:



Data Fields

- int **initialized**
- http_method_t method
- uri_type uri
- http_method_t request_method
- int status_code
- membuffer status_msg
- size_t amount_discarded
- int is_request
- int **major_version**
- int **minor_version**
- LinkedList headers
- memptr entity
- membuffer msg
- char * urlbuf

8.22.1 Field Documentation

8.22.1.1 amount_discarded

```
size_t http_message_t::amount_discarded
```

response only. the amount of data that's been read by the user, that's no longer in the raw message buffer.

Referenced by http_ReadHttpResponse().

8.22.1.2 entity

```
memptr http_message_t::entity
```

message body(entity).

Referenced by gena_process_notification_event(), http_ReadHttpResponse(), http_RecvPostMessage(), and soap_device_callback().

8.22.1.3 headers

```
LinkedList http_message_t::headers
```

Referenced by CheckOtherHTTPHeaders(), ExtraHTTPHeaders(), http_GetHttpResponse(), and print_http_headers().

8.22.1.4 is_request

```
int http_message_t::is_request
```

if 1, msg is a request, else response.

Referenced by print_http_headers(), and ssdp_handle_ctrlpt_msg().

8.22.1.5 method

```
http_method_t http_message_t::method
```

request only.

Referenced by check_soapaction_hdr(), genaCallback(), get_mpost_acton_hdrval(), print_http_headers(), process_request(), ssdp_event_handler_thread(), start_event_handler(), and valid_ssdp_msg().

8.22.1.6 msg

```
membuffer http_message_t::msg
```

entire raw message.

Referenced by gena_process_notification_event(), http_ReadHttpResponse(), http_RecvMessage(), http_RecvPostMessage(), readFromSSDPSocket(), and ReadResponseLineAndHeaders().

8.22.1.7 request_method

```
http_method_t http_message_t::request_method
```

response only.

Referenced by ssdp_event_handler_thread(), and valid_ssdp_msg().

8.22.1.8 status_code

```
int http_message_t::status_code
```

response only.

Referenced by gena_subscribe(), gena_unsubscribe(), genaNotify(), http_GetHttpResponse(), print_http_headers(), and ssdp_handle_ctrlpt_msg().

8.22.1.9 status_msg

```
membuffer http_message_t::status_msg
```

response only.

Referenced by print_http_headers().

8.22.1.10 uri

```
uri_type http_message_t::uri
```

request only.

Referenced by gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_re get_dev_service(), print_http_headers(), process_request(), and valid_ssdp_msg().

8.22.1.11 urlbuf

```
char* http_message_t::urlbuf
```

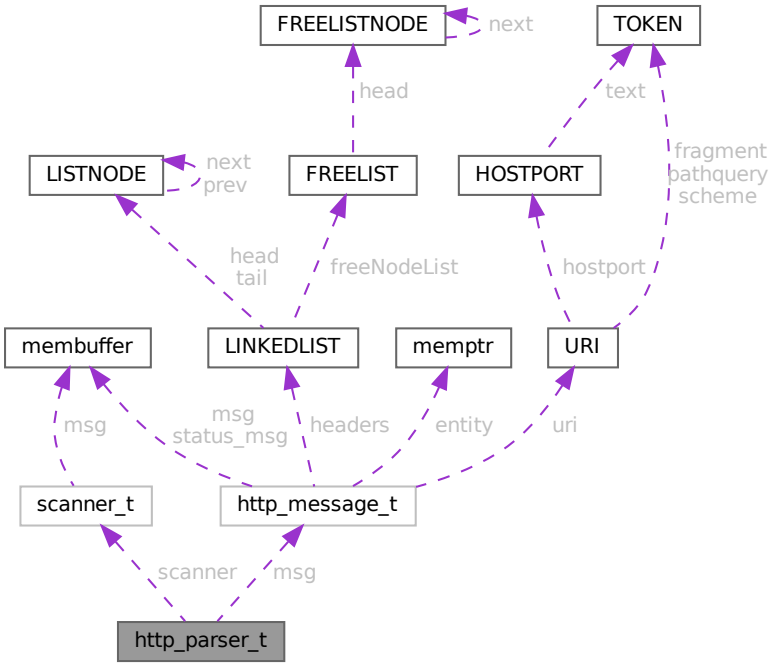
storage for url string.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h

8.23 http_parser_t Struct Reference

Collaboration diagram for http_parser_t:



Data Fields

- http_message_t **msg**
- int http_error_code
- int valid_ssdp_notify_hack
- parser_pos_t **position**
- int **ent_position**
- unsigned int **content_length**
- size_t **chunk_size**
- size_t entity_start_position
- scanner_t **scanner**

8.23.1 Field Documentation

8.23.1.1 entity_start_position

```
size_t http_parser_t::entity_start_position
```

offset in the the raw message buffer, which contains the message body. preceding this are the headers of the messsage.

Referenced by http_ReadHttpResponse(), and http_RecvPostMessage().

8.23.1.2 http_error_code

```
int http_parser_t::http_error_code
```

read-only; in case of parse error, this contains the HTTP error code (4XX or 5XX).

Referenced by http_ReadHttpResponse(), http_RecvMessage(), http_RecvPostMessage(), and ReadResponseLineAndHeaders().

8.23.1.3 valid_ssdp_notify_hack

```
int http_parser_t::valid_ssdp_notify_hack
```

read-only; this is set to 1 if a NOTIFY request has no content-length. used to read valid sspd notify msg.

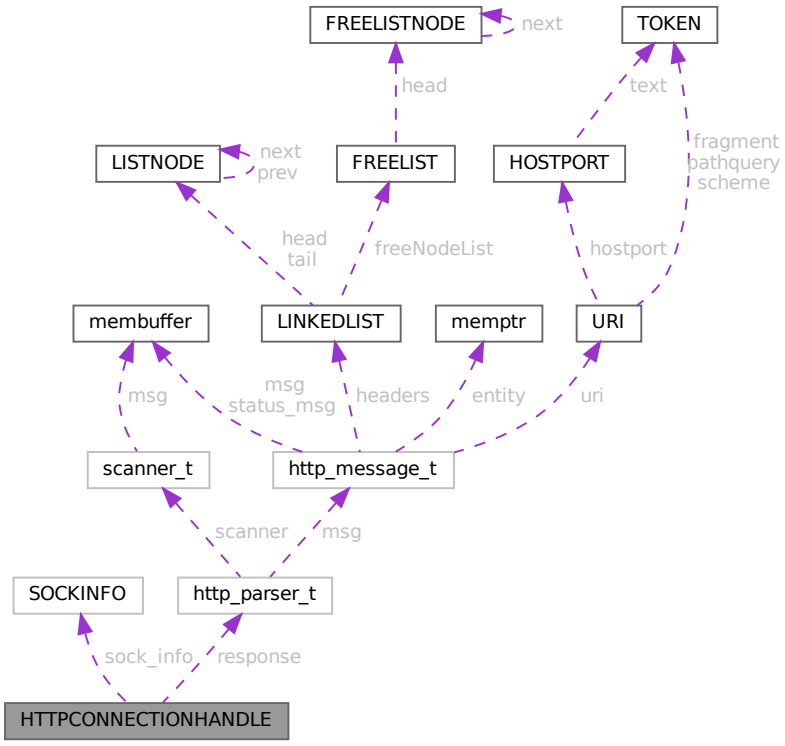
Referenced by start_event_handler().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h

8.24 HTTPCONNECTIONHANDLE Struct Reference

Collaboration diagram for HTTPCONNECTIONHANDLE:



Data Fields

- SOCKINFO **sock_info**
- int **contentLength**
- http_parser_t **response**
- int **requestStarted**
- int **cancel**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/httpreadwrite.c

8.25 ixml_membuf Struct Reference

The ixml_membuf type.

`#include <ixmlmembuf.h>`

Data Fields

- char * **buf**
- size_t **length**
- size_t **capacity**
- size_t **size_inc**

8.25.1 Detailed Description

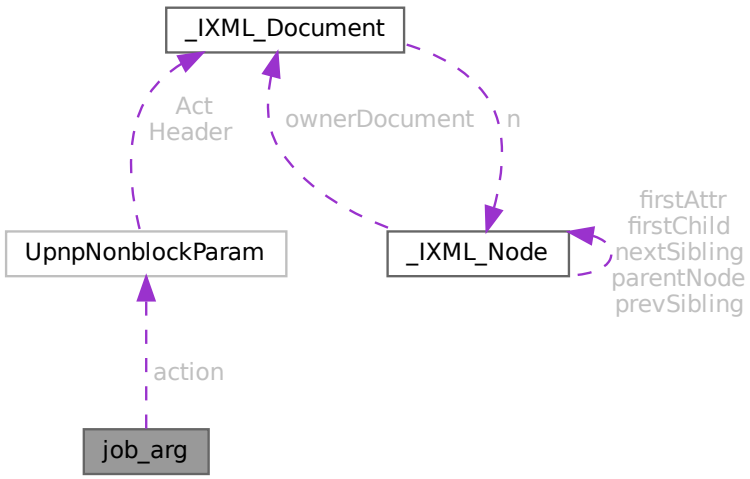
The ixml_membuf type.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlmembuf.h

8.26 job_arg Struct Reference

Collaboration diagram for job_arg:



Data Fields

- struct {
  int **handle**
  int **eventId**
  void * **Event**
  } **advertise**

- struct UpnpNonblockParam **action**

8.26.1 Detailed Description

Global variable used as to store the OpenSSL context object to be used for all SSL/TLS connections

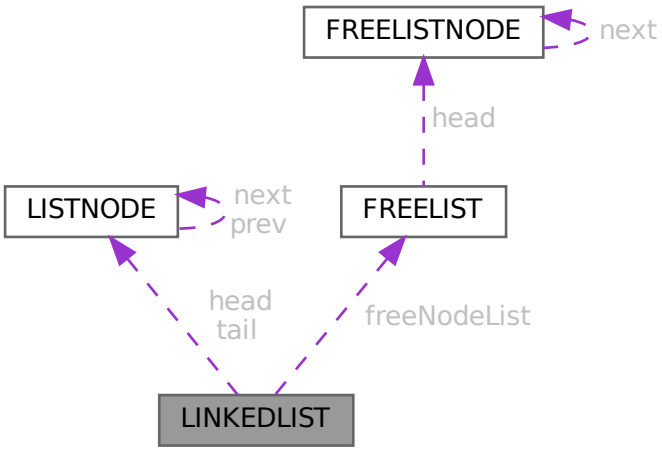
The documentation for this struct was generated from the following files:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpapi.c
- /build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_ctrlpt.c

8.27 LINKEDLIST Struct Reference

`#include <LinkedList.h>`

Collaboration diagram for LINKEDLIST:



Data Fields

- ListNode head
- ListNode tail
- long size
- FreeList freeNodeList
- free_function free_func
- cmp_routine cmp_func

8.27.1 Detailed Description

Linked list (no protection).

Because this is for internal use, parameters are NOT checked for validity. The first item of the list is stored at node: head->next The last item of the list is stored at node: tail->prev If head->next=tail, then list is empty. To iterate through the list:

```
LinkedList g;
ListNode *temp = NULL;
for (temp = ListHead(g);temp!=NULL;temp = ListNext(g,temp)) {
}
```

8.27.2 Field Documentation

8.27.2.1 cmp_func

```
cmp_routine LINKEDLIST::cmp_func
```

compare function to use

Referenced by ListFind(), and ListInit().

8.27.2.2 free_func

```
free_function LINKEDLIST::free_func
```

free function to use

Referenced by ListDelNode(), and ListInit().

8.27.2.3 freeNodeList

```
FreeList LINKEDLIST::freeNodeList
```

free list to use

Referenced by ListDestroy(), and ListInit().

8.27.2.4 head

```
ListNode LINKEDLIST::head
```

head, first item is stored at: head->next

Referenced by BumpPriority(), ListAddHead(), ListDelNode(), ListDestroy(), ListFind(), ListHead(), ListInit(), and ListPrev().

8.27.2.5 size

```
long LINKEDLIST::size
```

size of list

Referenced by AddWorker(), BumpPriority(), ListAddAfter(), ListAddBefore(), ListDelNode(), ListDestroy(), ListHead(), ListInit(), ListSize(), ListTail(), ThreadPoolAdd(), ThreadPoolShutdown(), TimerThreadWorker(), and WorkerThread().

8.27.2.6 tail

```
ListNode LINKEDLIST::tail
```

tail, last item is stored at: tail->prev

Referenced by ListAddTail(), ListDelNode(), ListDestroy(), ListFind(), ListInit(), ListNext(), and ListTail().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/LinkedList.h

8.28 LISTNODE Struct Reference

```
#include <LinkedList.h>
```

Collaboration diagram for LISTNODE:

Data Fields

- struct LISTNODE * **prev**
- struct LISTNODE * **next**
- void * **item**

8.28.1 Detailed Description

Linked list node. Stores generic item and pointers to next and prev.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/LinkedList.h

8.29 MD5Context Struct Reference

Data Fields

- uint32_t **state** [4]
- uint64_t **count**
- uint8_t **buffer** [64]

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/md5.h

8.30 membuffer Struct Reference

```
#include <membuffer.h>
```

Data Fields

- char * buf
- size_t length
- size_t capacity
- size_t size_inc

8.30.1 Detailed Description

Maintains a block of dynamically allocated memory note: Total length/capacity should not exceed MAX_INT

8.30.2 Field Documentation

8.30.2.1 buf

```
char* membuffer::buf
```

mem buffer; must not write beyond buf[length-1] (read/write).

Referenced by Check_Connect_And_Wait_Connection(), CheckOtherHTTPHeaders(), ExtraHTTPHeaders(), gena_process_subscription_renewal_request(), gena_process_unsubscribe_request(), gena_subscribe(), gena_unsubscribe(), get_alias(), get_mpost_acton_hdrval(), http_MakeHttpRequest(), http_ReadHttpResponse(), http_RecvMessage(), http_RecvPostMessage(), is_valid_alias(), membuffer_assign(), membuffer_attach(), membuffer_delete(), membuffer_destroy(), membuffer_detach(), membuffer_insert(), membuffer_set_size(), notify_send_and_recv(), print_http_headers(), process_request(), readFromSSDPSocket(), respond_ok(), send_action_response(), send_error_response(), send_var_query_response(), web_server_callback(), and web_server_set_root_dir().

8.30.2.2 capacity

`size_t membuffer::capacity`

total allocated memory (read-only).

Referenced by membuffer_attach(), and membuffer_set_size().

8.30.2.3 length

`size_t membuffer::length`

length of buffer (read-only).

Referenced by Check_Connect_And_Wait_Connection(), CheckOtherHTTPHeaders(), ExtraHTTPHeaders(), gena_process_notification_event(), gena_subscribe(), gena_unsubscribe(), get_alias(), get_mpost_acton_hdrval(), http_MakeHttpRequest(), membuffer_append(), membuffer_append_str(), membuffer_assign(), membuffer_attach(), membuffer_delete(), membuffer_insert(), membuffer_set_size(), notify_send_and_recv(), print_http_headers(), process_request(), readFromSSDPSocket(), respond_ok(), send_action_response(), send_error_response(), send_var_query_response(), web_server_callback(), and web_server_set_root_dir().

8.30.2.4 size_inc

`size_t membuffer::size_inc`

used to increase size; MUST be $>$ 0; (read/write).

Referenced by CreateServicePacket(), gena_subscribe(), gena_unsubscribe(), membuffer_init(), membuffer_set_size(), and respond_ok().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/membuffer.h

8.31 memptr Struct Reference

`#include <membuffer.h>`

Data Fields

- char * buf
- size_t length

8.31.1 Detailed Description

pointer to a chunk of memory.

8.31.2 Field Documentation

8.31.2.1 buf

```
char* memptr::buf
```

start of memory (read/write).

Referenced by Check_Connect_And_Wait_Connection(), check_soap_request(), check_soapaction_hdr(), CheckOtherHTTPHeaders(), create_url_list(), ExtraHTTPHeaders(), gena_process_notification_event(), gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_request(), gena_subscribe(), get_mpost_acton_hdrval(), handle_invoke_action(), has_xml_content_type(), http_GetHttpResponse(), memptr_cmp(), memptr_cmp_nocase(), print_http_headers(), process_request(), soap_device_callback(), ssdp_handle_ctrlpt_msg(), and ssdp_handle_device_request().

8.31.2.2 length

```
size_t memptr::length
```

length of memory (read-only).

Referenced by Check_Connect_And_Wait_Connection(), check_soapaction_hdr(), CheckOtherHTTPHeaders(), create_url_list(), ExtraHTTPHeaders(), gena_process_notification_event(), gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_request(), gena_subscribe(), get_mpost_acton_hdrval(), handle_invoke_action(), has_xml_content_type(), http_ReadHttpResponse(), http_RecvPostMessage(), memptr_cmp(), memptr_cmp_nocase(), print_http_headers(), process_request(), ssdp_handle_ctrlpt_msg(), and ssdp_handle_device_request().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/membuffer.h

8.32 mserv_request_t Struct Reference

Data Fields

- SOCKET connfd
- struct sockaddr_storage foreign_sockaddr

8.32.1 Field Documentation

8.32.1.1 connfd

```
SOCKET mserv_request_t::connfd
```

Connection handle.

8.32.1.2 foreign_sockaddr

```
struct sockaddr_storage mserv_request_t::foreign_sockaddr
```

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/miniserver/miniserver.c

8.33 MServerSockArray Struct Reference

Data Fields

- SOCKET miniServerSock4
- SOCKET miniServerSock6
- SOCKET miniServerSock6UlaGua
- SOCKET miniServerStopSock
- SOCKET ssdpSock4
- SOCKET ssdpSock6
- SOCKET ssdpSock6UlaGua
- uint16_t **stopPort**
- uint16_t **miniServerPort4**
- uint16_t **miniServerPort6**
- uint16_t **miniServerPort6UlaGua**
- SOCKET ssdpReqSock4
- SOCKET ssdpReqSock6

8.33.1 Field Documentation

8.33.1.1 miniServerSock4

```
SOCKET MServerSockArray::miniServerSock4
```

IPv4 socket for listening for miniserver requests.

Referenced by RunMiniServer(), and StartMiniServer().

8.33.1.2 miniServerSock6

```
SOCKET MServerSockArray::miniServerSock6
```

IPv6 LLA Socket for listening for miniserver requests.

Referenced by RunMiniServer(), and StartMiniServer().

8.33.1.3 miniServerSock6UlaGua

```
SOCKET MServerSockArray::miniServerSock6UlaGua
```

IPv6 ULA or GUA Socket for listening for miniserver requests.

Referenced by RunMiniServer(), and StartMiniServer().

8.33.1.4 miniServerStopSock

```
SOCKET MServerSockArray::miniServerStopSock
```

Socket for stopping miniserver

Referenced by get_miniserver_stopsock(), RunMiniServer(), and StartMiniServer().

8.33.1.5 ssdpReqSock4

```
SOCKET MServerSockArray::ssdpReqSock4
```

IPv4 SSDP socket for sending search requests and receiving search replies

Referenced by get_ssdp_sockets(), RunMiniServer(), and StartMiniServer().

8.33.1.6 ssdpReqSock6

```
SOCKET MServerSockArray::ssdpReqSock6
```

IPv6 SSDP socket for sending search requests and receiving search replies

Referenced by get_ssdp_sockets(), RunMiniServer(), and StartMiniServer().

8.33.1.7 ssdpSock4

```
SOCKET MServerSockArray::ssdpSock4
```

IPv4 SSDP Socket for incoming advertisments and search requests.

Referenced by get_ssdp_sockets(), RunMiniServer(), and StartMiniServer().

8.33.1.8 ssdpSock6

```
SOCKET MServerSockArray::ssdpSock6
```

IPv6 LLA SSDP Socket for incoming advertisments and search requests.

Referenced by get_ssdp_sockets(), RunMiniServer(), and StartMiniServer().

8.33.1.9 ssdpSock6UlaGua

```
SOCKET MServerSockArray::ssdpSock6UlaGua
```

IPv6 ULA or GUA SSDP Socket for incoming advertisments and search requests.

Referenced by get_ssdp_sockets(), RunMiniServer(), and StartMiniServer().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/miniserver.h

8.34 NOTIFY_THREAD_STRUCT Struct Reference

`#include <gena.h>`

Data Fields

- char * **headers**
- DOMString **propertySet**
- char * **servId**
- char * **UDN**
- Upnp_SID **sid**
- time_t **ctime**
- int * **reference_count**
- UpnpDevice_Handle **device_handle**

8.34.1 Detailed Description

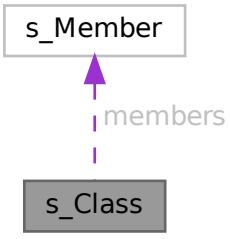
Structure to send NOTIFY message to all subscribed control points

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena.h

8.35 s_Class Struct Reference

Collaboration diagram for s_Class:



Data Fields

- const char * **name**
- unsigned **n_members**
- struct s_Member * **members**
- const char * **source**
- const char * **header**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/generator/generator.h

8.36 s_GenlibClientSubscription Struct Reference

Collaboration diagram for s_GenlibClientSubscription:

Data Fields

- int **m_RenewEventId**
- UpnpString * **m_SID**
- UpnpString * **m_ActualSID**
- UpnpString * **m_EventURL**
- GenlibClientSubscription * **m_Next**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/client_table/GenlibClientSubscription.c

8.37 s_Member Struct Reference

Data Fields

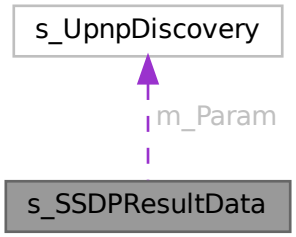
- const char * **name**
- enum e_Type **type**
- const char * **type_name**
- const char * **header**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/generator/generator.h

8.38 s_SSDPResultData Struct Reference

Collaboration diagram for s_SSDPResultData:



Data Fields

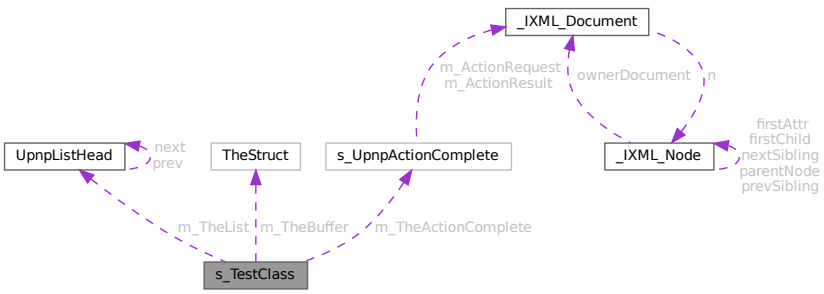
- UpnpDiscovery * **m_Param**
- void * **m_Cookie**
- Upnp_FunPtr **m_CtrlptCallback**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.c

8.39 s_TestClass Struct Reference

Collaboration diagram for s_TestClass:



Data Fields

- UpnpListHead **m_TheList**
- int **m_TheInteger**
- unsigned long **m_TheUnsignedLong**
- char * **m_TheCharPointer**
- struct TheStruct **m_TheBuffer**
- UpnpActionComplete * **m_TheActionComplete**
- UpnpString * **m_TheString**
- DOMString **m_TheDomString**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.c

8.40 s_Type_Integer Struct Reference

Data Fields

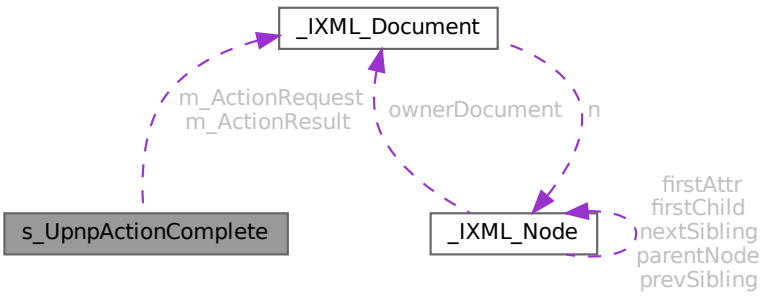
- const char * **int_type**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/generator/generator.h

8.41 s_UpnpActionComplete Struct Reference

Collaboration diagram for s_UpnpActionComplete:



Data Fields

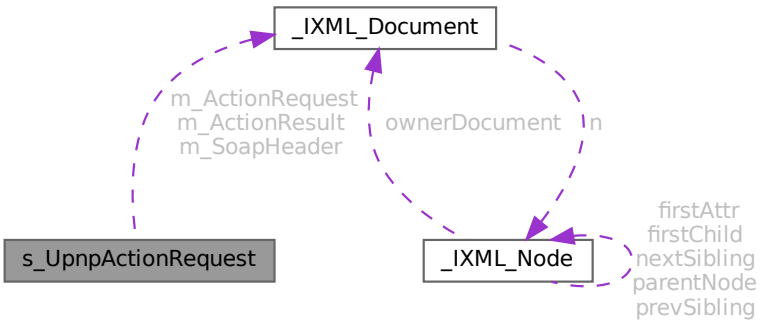
- int **m_ErrCode**
- UpnpString * **m_CtrlUrl**
- IXML_Document * **m_ActionRequest**
- IXML_Document * **m_ActionResult**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionComplete.c

8.42 s_UpnpActionRequest Struct Reference

Collaboration diagram for s_UpnpActionRequest:



Data Fields

- int **m_ErrCode**
- int **m_Socket**
- UpnpString * **m_ErrStr**
- UpnpString * **m_ActionName**
- UpnpString * **m_DevUDN**
- UpnpString * **m_ServiceID**
- IXML_Document * **m_ActionRequest**
- IXML_Document * **m_ActionResult**
- IXML_Document * **m_SoapHeader**
- struct sockaddr_storage **m_CtrlPtIPAddr**
- UpnpString * **m_Os**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionRequest.c

8.43 s_UpnpDiscovery Struct Reference

Data Fields

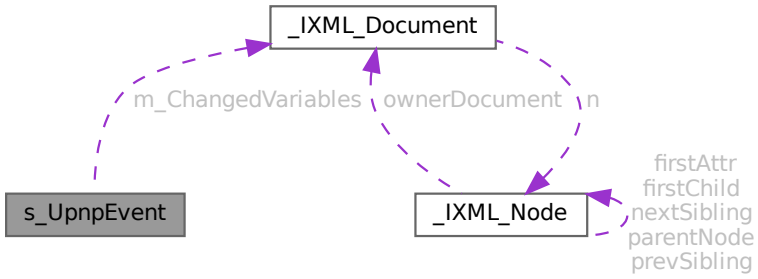
- int **m_ErrCode**
- int **m_Expires**
- UpnpString * **m_DeviceID**
- UpnpString * **m_DeviceType**
- UpnpString * **m_ServiceType**
- UpnpString * **m_ServiceVer**
- UpnpString * **m_Location**
- UpnpString * **m_Os**
- UpnpString * **m_Date**
- UpnpString * **m_Ext**
- struct sockaddr_storage **m_DestAddr**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpDiscovery.c

8.44 s_UpnpEvent Struct Reference

Collaboration diagram for s_UpnpEvent:



Data Fields

- int **m_EventKey**
- IXML_Document * **m_ChangedVariables**
- UpnpString * **m_SID**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEvent.c

8.45 s_UpnpEventSubscribe Struct Reference

Data Fields

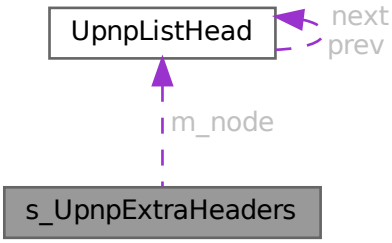
- int **m_ErrCode**
- int **m_TimeOut**
- UpnpString * **m_SID**
- UpnpString * **m_PublisherUrl**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEventSubscribe.c

8.46 s_UpnpExtraHeaders Struct Reference

Collaboration diagram for s_UpnpExtraHeaders:



Data Fields

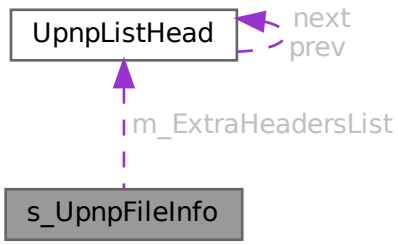
- UpnpListHead **m_node**
- UpnpString * **m_name**
- UpnpString * **m_value**
- DOMString **m_resp**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpExtraHeaders.c

8.47 s_UpnpFileInfo Struct Reference

Collaboration diagram for s_UpnpFileInfo:



Data Fields

- off_t **m_FileLength**
- time_t **m_LastModified**
- int **m_IsDirectory**
- int **m_IsReadable**
- DOMString **m_ContentType**
- UpnpListHead **m_ExtraHeadersList**
- struct sockaddr_storage **m_CtrlPtIPAddr**
- UpnpString * **m_Os**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpFileInfo.c

8.48 s_UpnpStateVarComplete Struct Reference

Data Fields

- int **m_ErrCode**
- UpnpString * **m_CtrlUrl**
- UpnpString * **m_StateVarName**
- DOMString **m_CurrentVal**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarComplete.c

8.49 s_UpnpStateVarRequest Struct Reference

Data Fields

- int **m_ErrCode**
- int **m_Socket**
- UpnpString * **m_ErrStr**
- UpnpString * **m_DevUDN**
- UpnpString * **m_ServiceID**
- UpnpString * **m_StateVarName**
- struct sockaddr_storage **m_CtrlPtIPAddr**
- DOMString **m_CurrentVal**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarRequest.c

8.50 s_UpnpSubscriptionRequest Struct Reference

Data Fields

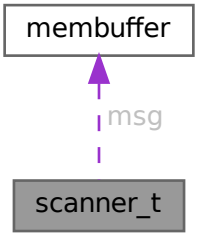
- UpnpString * **m_ServiceId**
- UpnpString * **m_UDN**
- UpnpString * **m_SID**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpSubscriptionRequest.c

8.51 scanner_t Struct Reference

Collaboration diagram for scanner_t:



Data Fields

- membuffer * msg
- size_t cursor
- int entire_msg_loaded

8.51.1 Field Documentation

8.51.1.1 cursor

```
size_t scanner_t::cursor
```

current position in buffer.

Referenced by http_ReadHttpResponse().

8.51.1.2 entire_msg_loaded

```
int scanner_t::entire_msg_loaded
```

set this to 1 if the entire msg is loaded in 'msg'; else 0 if only partial msg in 'msg' (default is 0).

8.51.1.3 msg

```
membuffer* scanner_t::msg
```

raw http msg.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h

8.52 SendInstruction Struct Reference

Data Fields

- int **IsVirtualFile**
- int **IsChunkActive**
- int **IsRangeActive**
- int **IsTrailers**
- char **RangeHeader** [200]
- char **AcceptLanguageHeader** [200]
- off_t **RangeOffset**
- off_t ReadSendSize
- long RecvWriteSize
- const void * CorsHeader
- const void * Cookie
- const void * RequestCookie

8.52.1 Field Documentation

8.52.1.1 Cookie

```
const void* SendInstruction::Cookie
```

Cookie associated with the virtualDir.

Referenced by http_RecvPostMessage(), http_SendMessage(), and process_request().

8.52.1.2 CorsHeader

```
const void* SendInstruction::CorsHeader
```

CorsHeader.

Referenced by http_MakeMessage(), and process_request().

8.52.1.3 ReadSendSize

```
off_t SendInstruction::ReadSendSize
```

Read from local source and send on the network.

Referenced by CreateHTTPRangeResponseHeader(), http_SendMessage(), and process_request().

8.52.1.4 RecvWriteSize

```
long SendInstruction::RecvWriteSize
```

Recv from the network and write into local file.

Referenced by CheckOtherHTTPHeaders().

8.52.1.5 RequestCookie

```
const void* SendInstruction::RequestCookie
```

Cookie associated with the request.

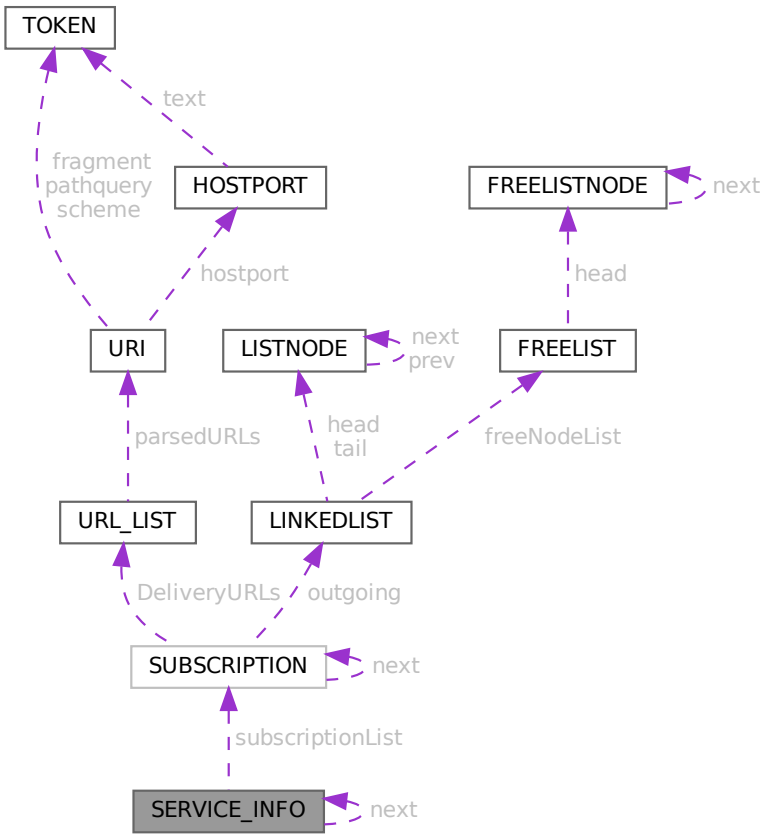
Referenced by http_RecvPostMessage(), http_SendMessage(), and process_request().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/webserver.h

8.53 SERVICE_INFO Struct Reference

Collaboration diagram for SERVICE_INFO:



Data Fields

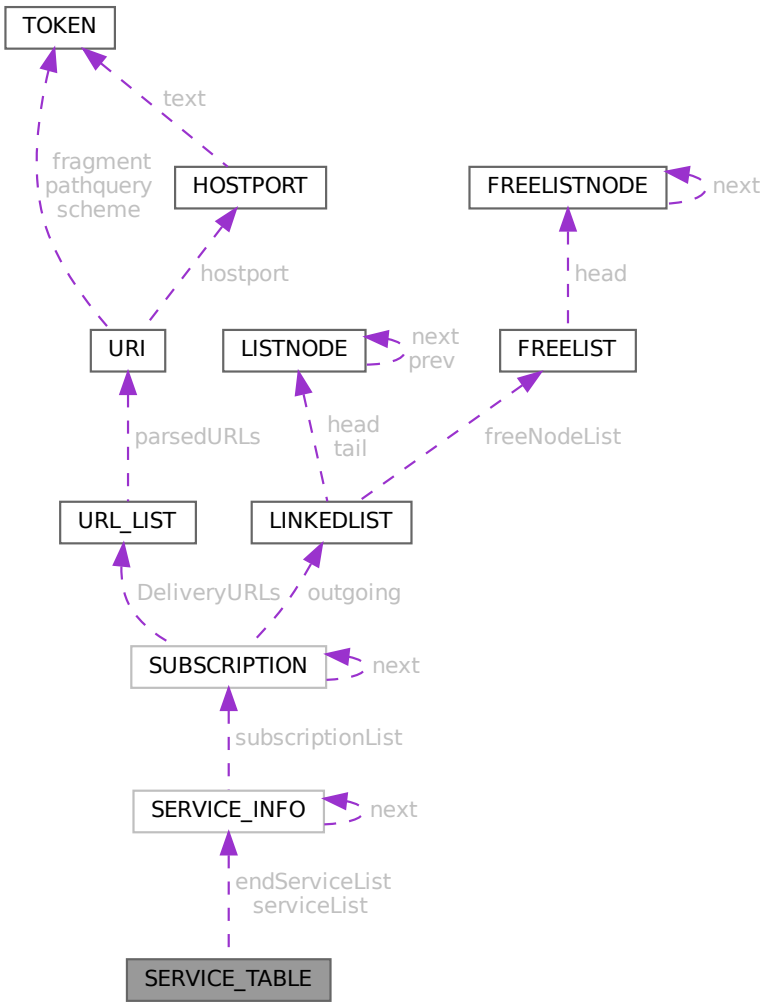
- DOMString **serviceType**
- DOMString **serviceId**
- char * **SCPDURL**
- char * **controlURL**
- char * **eventURL**
- DOMString **UDN**
- int **active**
- int **TotalSubscriptions**
- subscription * **subscriptionList**
- struct SERVICE_INFO * **next**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/service_table.h

8.54 SERVICE_TABLE Struct Reference

Collaboration diagram for SERVICE_TABLE:



Data Fields

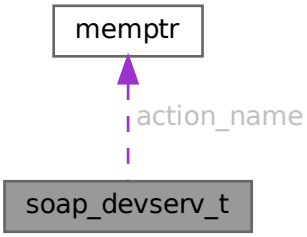
- DOMString **URLBase**
- service_info * **serviceList**
- service_info * **endServiceList**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/service_table.h

8.55 soap_devserv_t Struct Reference

Collaboration diagram for soap_devserv_t:



Data Fields

- char **dev_udn** [NAME_SIZE]
- char **service_type** [NAME_SIZE]
- char **service_id** [NAME_SIZE]
- memptr **action_name**
- Upnp_FunPtr **callback**
- void * **cookie**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/soap/soap_device.c

8.56 SOCKINFO Struct Reference

Data Fields

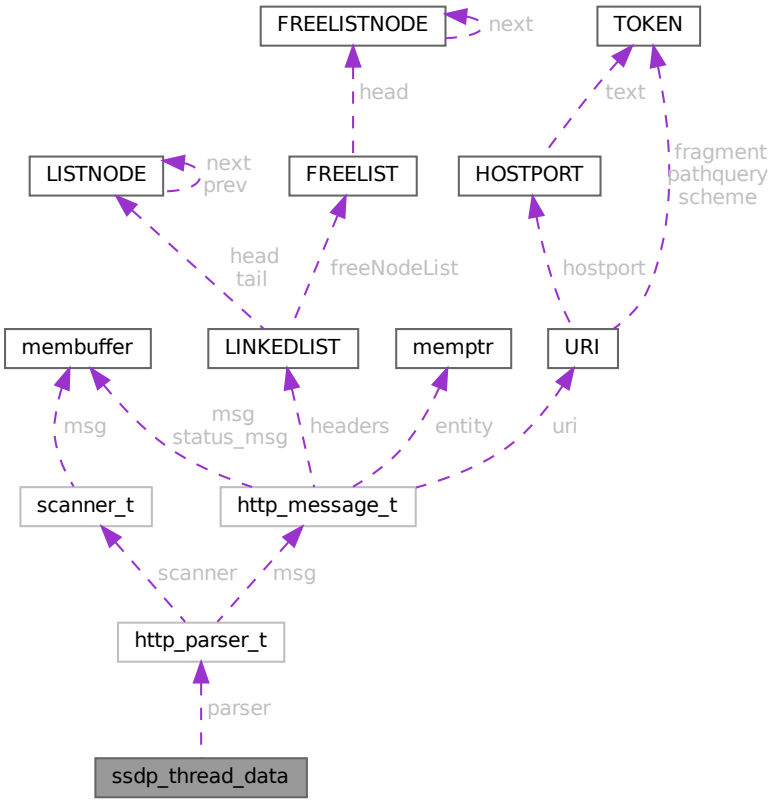
- SOCKET socket
- struct sockaddr_storage foreign_sockaddr

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sock.h

8.57 ssdp_thread_data Struct Reference

Collaboration diagram for ssdp_thread_data:



Data Fields

- http_parser_t **parser**
- struct sockaddr_storage **dest_addr**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h

8.58 SsdpEventStruct Struct Reference

```
#include <ssdplib.h>
```

Data Fields

- enum SsdpSearchType **RequestType**
- int **ErrCode**
- int **MaxAge**
- int **Mx**
- char **UDN** [LINE_SIZE]
- char **DeviceType** [LINE_SIZE]
- char **ServiceType** [LINE_SIZE]
- char **Location** [LINE_SIZE]
- char **HostAddr** [LINE_SIZE]
- char **Os** [LINE_SIZE]
- char **Ext** [LINE_SIZE]
- char **Date** [LINE_SIZE]
- struct sockaddr * **DestAddr**
- void * **Cookie**

8.58.1 Detailed Description

Structure to store the SSDP information

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h

8.59 ssdpsearcharg Struct Reference

Data Fields

- int **timeoutEventId**
- char * **searchTarget**
- void * **cookie**
- enum SsdpSearchType **requestType**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h

8.60 ssdpsearchexparg Struct Reference

Data Fields

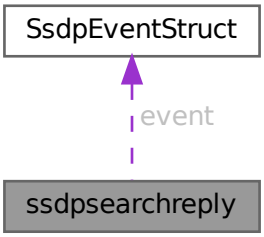
- int **handle**
- int **timeoutEventId**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h

8.61 ssdpsearchreply Struct Reference

Collaboration diagram for ssdpsearchreply:



Data Fields

- int **MaxAge**
- UpnpDevice_Handle **handle**
- struct sockaddr_storage **dest_addr**
- SsdpEvent **event**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h

8.62 SSDPSockArray Struct Reference

Data Fields

- SOCKET ssdpSock
- int ssdpReqSock

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_server.c

8.63 str_int_entry Struct Reference

Data Fields

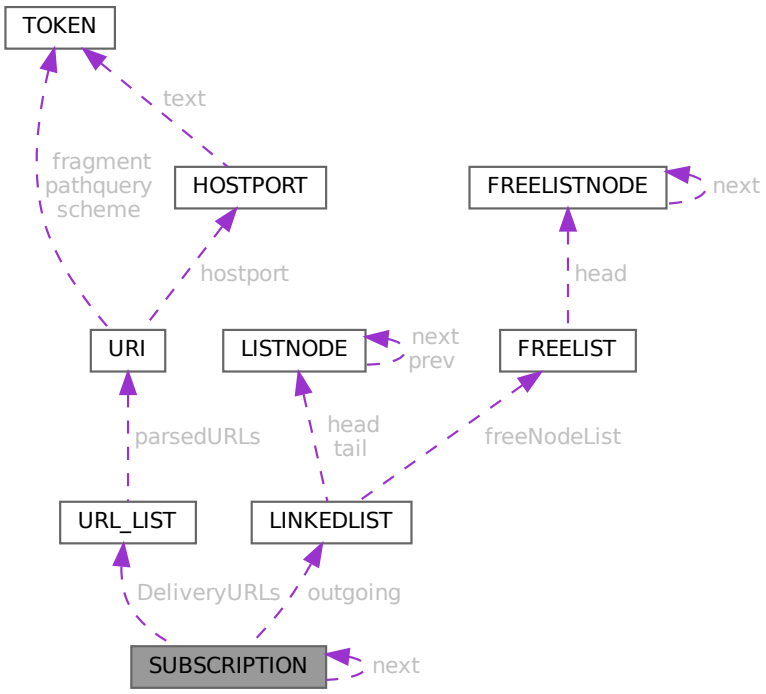
- const char * **name**
- int **id**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/strintmap.h

8.64 SUBSCRIPTION Struct Reference

Collaboration diagram for SUBSCRIPTION:



Data Fields

- Upnp_SID **sid**
- int **ToSendEventKey**
- time_t **expireTime**
- int **active**
- URL_list **DeliveryURLs**
- LinkedList **outgoing**
- struct SUBSCRIPTION * **next**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/service_table.h

8.65 SUpnpString Struct Reference

Internal implementation of the class UpnpString.

Data Fields

- size_t m_length

  Length of the string.
- char * m_string

  Pointer to a dynamically allocated area that holds the NULL terminated string.

8.65.1 Detailed Description

Internal implementation of the class UpnpString.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpString.c

8.66 TData Struct Reference

Data Fields

- int **Mx**
- void * **Cookie**
- char * **Data**
- struct sockaddr_storage **DestAddr**

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h

8.67 TheStruct Struct Reference

Data Fields

- int **x**
- int **y**

The documentation for this struct was generated from the following file:

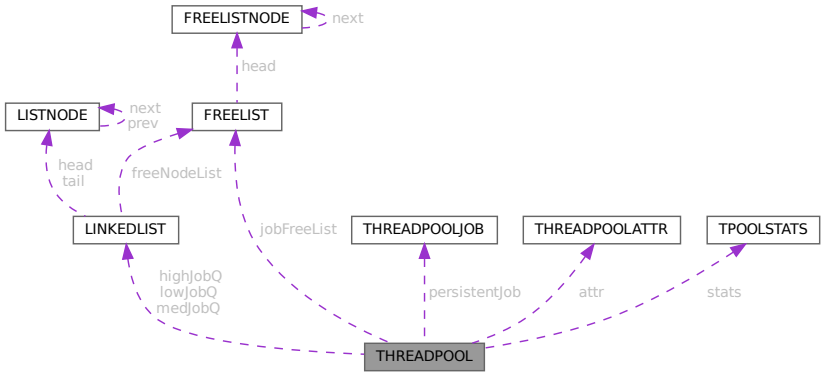
- /build/reproducible-path/pupnp-1.14.20/upnp/generator/TheStruct.h

8.68 THREADPOOL Struct Reference

A thread pool similar to the thread pool in the UPnP SDK.

`#include <ThreadPool.h>`

Collaboration diagram for THREADPOOL:



Data Fields

- ithread_mutex_t mutex
- ithread_cond_t condition
- ithread_cond_t start_and_shutdown
- int lastJobId
- int shutdown
- int totalThreads
- int pendingWorkerThreadStart
- int busyThreads
- int persistentThreads
- FreeList jobFreeList
- LinkedList lowJobQ
- LinkedList medJobQ
- LinkedList highJobQ
- ThreadPoolJob * persistentJob
- ThreadPoolAttr attr
- ThreadPoolStats stats

8.68.1 Detailed Description

A thread pool similar to the thread pool in the UPnP SDK.

Allows jobs to be scheduled for running by threads in a thread pool. The thread pool is initialized with a minimum and maximum thread number as well as a max idle time and a jobs per thread ratio. If a worker thread waits the whole max idle time without receiving a job and the thread pool currently has more threads running than the minimum then the worker thread will exit. If when scheduling a job the current job to thread ratio becomes greater than the set ratio and the thread pool currently has less than the maximum threads then a new thread will be created.

8.68.2 Field Documentation

8.68.2.1 attr

```
ThreadPoolAttr THREADPOOL::attr
```

thread pool attributes

Referenced by AddWorker(), BumpPriority(), CreateWorker(), ThreadPoolAdd(), ThreadPoolAddPersistent(), ThreadPoolGetAttr(), ThreadPoolInit(), ThreadPoolSetAttr(), and WorkerThread().

8.68.2.2 busyThreads

```
int THREADPOOL::busyThreads
```

number of threads that are currently executing jobs

Referenced by AddWorker(), ThreadPoolInit(), and WorkerThread().

8.68.2.3 condition

```
ithread_cond_t THREADPOOL::condition
```

Condition variable to signal Q.

Referenced by ThreadPoolAdd(), ThreadPoolAddPersistent(), ThreadPoolInit(), ThreadPoolSetAttr(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.4 highJobQ

```
LinkedList THREADPOOL::highJobQ
```

high priority job Q

Referenced by AddWorker(), BumpPriority(), ThreadPoolAdd(), ThreadPoolGetStats(), ThreadPoolInit(), ThreadPoolRemove(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.5 jobFreeList

```
FreeList THREADPOOL::jobFreeList
```

free list of jobs

Referenced by CreateThreadPoolJob(), FreeThreadPoolJob(), ThreadPoolInit(), and ThreadPoolShutdown().

8.68.2.6 lastJobId

```
int THREADPOOL::lastJobId
```

ids for jobs

Referenced by ThreadPoolAdd(), ThreadPoolAddPersistent(), and ThreadPoolInit().

8.68.2.7 lowJobQ

```
LinkedList THREADPOOL::lowJobQ
```

low priority job Q

Referenced by AddWorker(), BumpPriority(), ThreadPoolAdd(), ThreadPoolGetStats(), ThreadPoolInit(), ThreadPoolRemove(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.8 medJobQ

```
LinkedList THREADPOOL::medJobQ
```

med priority job Q

Referenced by AddWorker(), BumpPriority(), ThreadPoolAdd(), ThreadPoolGetStats(), ThreadPoolInit(), ThreadPoolRemove(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.9 mutex

```
ithread_mutex_t THREADPOOL::mutex
```

Mutex to protect job qs.

Referenced by CreateWorker(), ThreadPoolAdd(), ThreadPoolAddPersistent(), ThreadPoolGetAttr(), ThreadPoolGetStats(), ThreadPoolInit(), ThreadPoolRemove(), ThreadPoolSetAttr(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.10 pendingWorkerThreadStart

```
int THREADPOOL::pendingWorkerThreadStart
```

flag that's set when waiting for a new worker thread to start

Referenced by CreateWorker(), ThreadPoolInit(), and WorkerThread().

8.68.2.11 persistentJob

```
ThreadPoolJob* THREADPOOL::persistentJob
```

persistent job

Referenced by ThreadPoolAddPersistent(), ThreadPoolInit(), ThreadPoolRemove(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.12 persistentThreads

```
int THREADPOOL::persistentThreads
```

number of persistent threads

Referenced by AddWorker(), ThreadPoolAddPersistent(), ThreadPoolGetStats(), ThreadPoolInit(), and WorkerThread().

8.68.2.13 shutdown

```
int THREADPOOL::shutdown
```

whether or not we are shutting down

Referenced by ThreadPoolGetAttr(), ThreadPoolGetStats(), ThreadPoolInit(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.14 start_and_shutdown

```
ithread_cond_t THREADPOOL::start_and_shutdown
```

Condition variable for start and stop.

Referenced by CreateWorker(), ThreadPoolAddPersistent(), ThreadPoolInit(), ThreadPoolShutdown(), and WorkerThread().

8.68.2.15 stats

```
ThreadPoolStats THREADPOOL::stats
```

statistics

Referenced by CreateWorker(), StatsAccountHQ(), StatsAccountLQ(), StatsAccountMQ(), ThreadPoolGetStats(), ThreadPoolInit(), and WorkerThread().

8.68.2.16 totalThreads

```
int THREADPOOL::totalThreads
```

total number of threads

Referenced by AddWorker(), CreateWorker(), ThreadPoolAddPersistent(), ThreadPoolGetStats(), ThreadPoolInit(), ThreadPoolSetAttr(), ThreadPoolShutdown(), and WorkerThread().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.h

8.69 THREADPOOLATTR Struct Reference

`#include <ThreadPool.h>`

Data Fields

- int minThreads
- int maxThreads
- size_t stackSize
- int maxIdleTime
- int jobsPerThread
- int maxJobsTotal
- int starvationTime
- PolicyType schedPolicy

8.69.1 Detailed Description

Attributes for thread pool. Used to set and change parameters of thread pool.

8.69.2 Field Documentation

8.69.2.1 jobsPerThread

`int THREADPOOLATTR::jobsPerThread`

Jobs per thread to maintain.

Referenced by AddWorker(), TPAttrInit(), and TPAttrSetJobsPerThread().

8.69.2.2 maxIdleTime

`int THREADPOOLATTR::maxIdleTime`

This is the maximum time a thread will remain idle before dying (in milliseconds).

Referenced by BumpPriority(), TPAttrInit(), TPAttrSetIdleTime(), and WorkerThread().

8.69.2.3 maxJobsTotal

`int THREADPOOLATTR::maxJobsTotal`

Maximum number of jobs that can be queued totally.

Referenced by ThreadPoolAdd(), TPAttrInit(), and TPAttrSetMaxJobsTotal().

8.69.2.4 maxThreads

```
int THREADPOOLATTR::maxThreads
```

ThreadPool will never have more than this number of threads.

Referenced by CreateWorker(), ThreadPoolAddPersistent(), TPAttrInit(), TPAttrSetMaxThreads(), and WorkerThread().

8.69.2.5 minThreads

```
int THREADPOOLATTR::minThreads
```

ThreadPool will always maintain at least this many threads.

Referenced by ThreadPoolInit(), ThreadPoolSetAttr(), TPAttrInit(), TPAttrSetMinThreads(), and WorkerThread().

8.69.2.6 schedPolicy

```
PolicyType THREADPOOLATTR::schedPolicy
```

scheduling policy to use.

Referenced by ThreadPoolInit(), ThreadPoolSetAttr(), TPAttrInit(), and TPAttrSetSchedPolicy().

8.69.2.7 stackSize

```
size_t THREADPOOLATTR::stackSize
```

This is the minimum stack size allocated for each thread.

Referenced by CreateWorker(), TPAttrInit(), and TPAttrSetStackSize().

8.69.2.8 starvationTime

```
int THREADPOOLATTR::starvationTime
```

the time a low priority or med priority job waits before getting bumped up a priority (in milliseconds).

Referenced by BumpPriority(), TPAttrInit(), and TPAttrSetStarvationTime().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.h

8.70 THREADPOOLJOB Struct Reference

```
#include <ThreadPool.h>
```

Data Fields

- start_routine **func**
- void * **arg**
- free_routine **free_func**
- struct timeval **requestTime**
- ThreadPriority **priority**
- int **jobId**

8.70.1 Detailed Description

Internal ThreadPool Job.

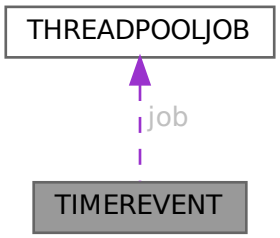
The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.h

8.71 TIMEREVENT Struct Reference

`#include <TimerThread.h>`

Collaboration diagram for TIMEREVENT:



Data Fields

- ThreadPoolJob **job**
- time_t eventTime
- Duration persistent
- int **id**

8.71.1 Detailed Description

Struct to contain information for a timer event.

Internal to the TimerThread.

8.71.2 Field Documentation

8.71.2.1 eventTime

```
time_t TIMEREVENT::eventTime
```

[in] Absolute time for event in seconds since Jan 1, 1970.

Referenced by CreateTimerEvent(), TimerThreadSchedule(), and TimerThreadWorker().

8.71.2.2 persistent

```
Duration TIMEREVENT::persistent
```

[in] Long term or short term job.

Referenced by CreateTimerEvent(), and TimerThreadWorker().

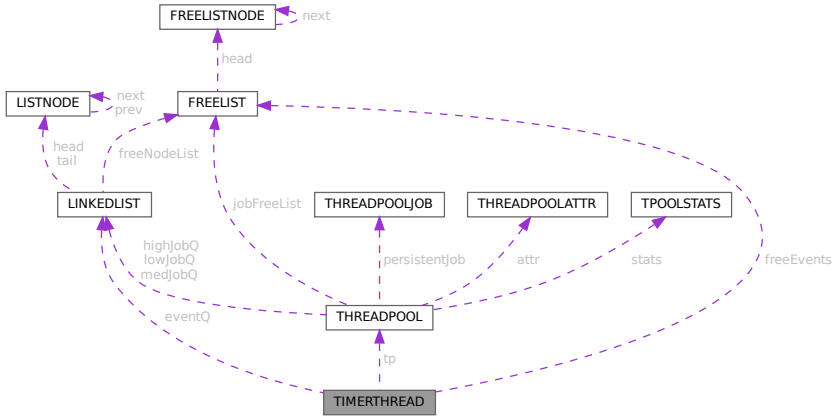
The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.h

8.72 TIMERTHREAD Struct Reference

```
#include <TimerThread.h>
```

Collaboration diagram for TIMERTHREAD:



Data Fields

- ithread_mutex_t **mutex**
- ithread_cond_t **condition**
- int **lastEventId**
- LinkedList **eventQ**
- int **shutdown**
- FreeList **freeEvents**
- ThreadPool * **tp**

8.72.1 Detailed Description

A timer thread similar to the one in the Upnp SDK that allows the scheduling of a job to run at a specified time in the future.

Because the timer thread uses the thread pool there is no gurantee of timing, only approximate timing.

Uses ThreadPool, Mutex, Condition, Thread.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.h

8.73 TOKEN Struct Reference

Buffer used in parsinghttp messages, urls, etc. generally this simply holds a pointer into a larger array.

`#include <uri.h>`

Data Fields

- const char * **buff**
- size_t **size**

8.73.1 Detailed Description

Buffer used in parsinghttp messages, urls, etc. generally this simply holds a pointer into a larger array.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h

8.74 TPOOLSTATS Struct Reference

`#include <ThreadPool.h>`

Data Fields

- double **totalTimeHQ**
- int **totalJobsHQ**
- double **avgWaitHQ**
- double **totalTimeMQ**
- int **totalJobsMQ**
- double **avgWaitMQ**
- double **totalTimeLQ**
- int **totalJobsLQ**
- double **avgWaitLQ**
- double **totalWorkTime**
- double **totalIdleTime**
- int **workerThreads**
- int **idleThreads**
- int **persistentThreads**
- int **totalThreads**
- int **maxThreads**
- int **currentJobsHQ**
- int **currentJobsLQ**
- int **currentJobsMQ**

8.74.1 Detailed Description

Structure to hold statistics.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.h

8.75 tv_service Struct Reference

Data Fields

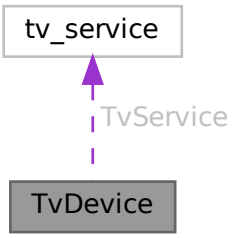
- char **ServiceId** [NAME_SIZE]
- char **ServiceType** [NAME_SIZE]
- char * **VariableStrVal** [4]
- char **EventURL** [NAME_SIZE]
- char **ControlURL** [NAME_SIZE]
- char **SID** [NAME_SIZE]

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.h

8.76 TvDevice Struct Reference

Collaboration diagram for TvDevice:



Data Fields

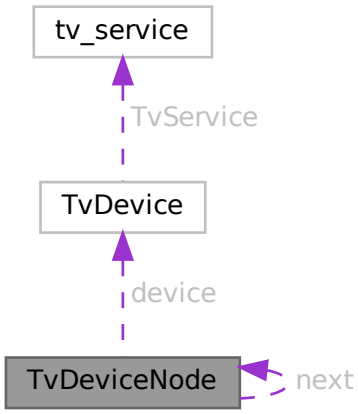
- char **UDN** [250]
- char **DescDocURL** [250]
- char **FriendlyName** [250]
- char **PresURL** [250]
- int **AdvrTimeOut**
- struct tv_service **TvService** [2]

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.h

8.77 TvDeviceNode Struct Reference

Collaboration diagram for TvDeviceNode:



Data Fields

- struct TvDevice **device**
- struct TvDeviceNode * **next**

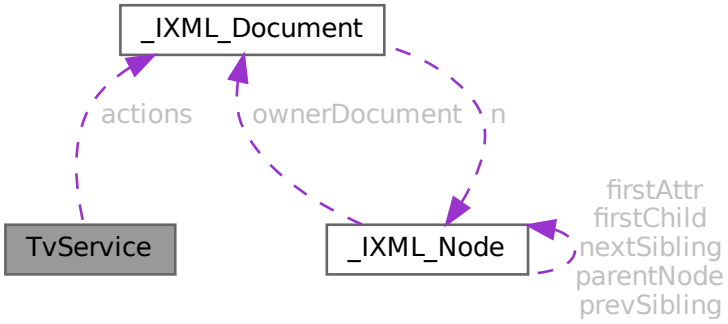
The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.h

8.78 TvService Struct Reference

`#include <tv_device.h>`

Collaboration diagram for TvService:



Data Fields

- char UDN [NAME_SIZE]
- char ServiceId [NAME_SIZE]
- char ServiceType [NAME_SIZE]
- const char * VariableName [4]
- char * VariableStrVal [4]
- const char * ActionNames [12]
- upnp_action actions [12]
- int VariableCount

8.78.1 Detailed Description

Structure for storing Tv Service identifiers and state table.

8.78.2 Field Documentation

8.78.2.1 ActionNames

```
const char* TvService::ActionNames[12]
```

Referenced by SetActionTable(), and TvDeviceHandleActionRequest().

8.78.2.2 actions

```
upnp_action TvService::actions[12]
```

Referenced by SetActionTable(), and TvDeviceHandleActionRequest().

8.78.2.3 ServiceId

```
char TvService::ServiceId[NAME_SIZE]
```

Referenced by SetServiceTable(), TvDeviceHandleActionRequest(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionR and TvDeviceSetServiceTableVar().

8.78.2.4 ServiceType

```
char TvService::ServiceType[NAME_SIZE]
```

Referenced by SetServiceTable().

8.78.2.5 UDN

```
char TvService::UDN[NAME_SIZE]
```

Universally Unique Device Name.

Referenced by SetServiceTable(), TvDeviceHandleActionRequest(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionR and TvDeviceSetServiceTableVar().

8.78.2.6 VariableCount

```
int TvService::VariableCount
```

Referenced by SetServiceTable(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), and TvDeviceSetServiceTableVar().

8.78.2.7 VariableName

```
const char* TvService::VariableName[ 4]
```

Referenced by SetServiceTable(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), and TvDeviceSetServiceTableVar().

8.78.2.8 VariableStrVal

```
char* TvService::VariableStrVal[ 4]
```

Referenced by IncrementBrightness(), IncrementColor(), IncrementContrast(), IncrementVolume(), SetServiceTable(), TvDeviceHandleActionRequest(), TvDeviceHandleGetVarRequest(), TvDeviceHandleSubscriptionRequest(), and TvDeviceSetServiceTableVar().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.h

8.79 UpnpListHead Struct Reference

```
#include <list.h>
```

Collaboration diagram for UpnpListHead:

Data Fields

- struct UpnpListHead * **next**
- struct UpnpListHead * **prev**

8.79.1 Detailed Description

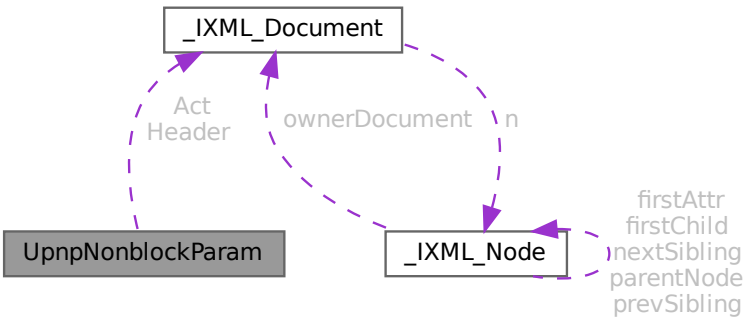
Trivial list management interface, patterned on std::list. It aims more at being familiar than at being minimal. The implementation does not perform any allocation or deallocation. List anchor structure. This should be the *first* entry in list member objects, except if you want to do member offset arithmetic instead of simple casts (look up "containerof"). The list code itself does not care.

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/inc/list.h

8.80 UpnpNonblockParam Struct Reference

Collaboration diagram for UpnpNonblockParam:



Data Fields

- UpnpFunName **FunName**
- int **Handle**
- int **TimeOut**
- char **VarName** [NAME_SIZE]
- char **NewVal** [NAME_SIZE]
- char **DevType** [NAME_SIZE]
- char **DevId** [NAME_SIZE]
- char **ServiceType** [NAME_SIZE]
- char **ServiceVer** [NAME_SIZE]
- char **Url** [NAME_SIZE]
- Upnp_SID **SubsId**
- char * **Cookie**
- Upnp_FunPtr **Fun**
- IXML_Document * **Header**
- IXML_Document * **Act**
- struct DevDesc * **Devdesc**

The documentation for this struct was generated from the following file:

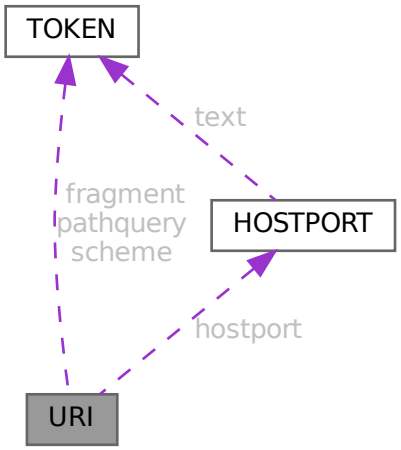
- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnpapi.h

8.81 URI Struct Reference

Represents a URI used in parse_uri and elsewhere.

`#include <uri.h>`

Collaboration diagram for URI:



Data Fields

- enum uriType **type**
- token **scheme**
- enum pathType **path_type**
- token **pathquery**
- token **fragment**
- hostport_type **hostport**

8.81.1 Detailed Description

Represents a URI used in parse_uri and elsewhere.

The documentation for this struct was generated from the following file:

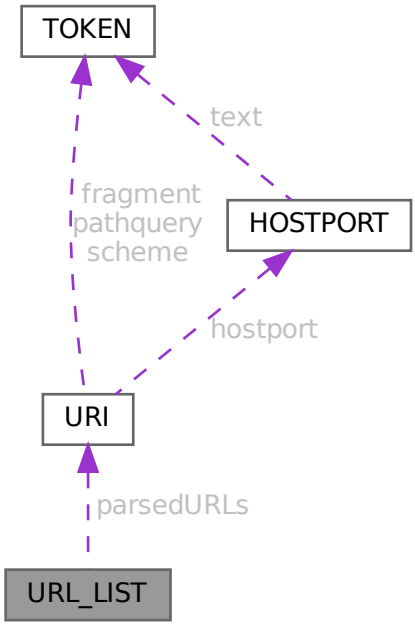
- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h

8.82 URL_LIST Struct Reference

Represents a list of URLs as in the "callback" header of SUBSCRIBE message in GENA. "char ∗" URLs holds dynamic memory.

`#include <uri.h>`

Collaboration diagram for URL_LIST:



Data Fields

- size_t **size**
- char ∗ URLs
- uri_type ∗ **parsedURLs**

8.82.1 Detailed Description

Represents a list of URLs as in the "callback" header of SUBSCRIBE message in GENA. "char ∗" URLs holds dynamic memory.

8.82.2 Field Documentation

8.82.2.1 URLs

```
char* URL_LIST::URLs
```

All the urls, delimited by <>

Referenced by copy_URL_list(), create_url_list(), free_URL_list(), and gena_process_subscription_request().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h

8.83 uuid_node_t Struct Reference

Data Fields

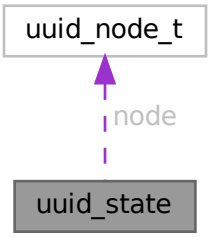
- char **nodeID** [6]

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sysdep.h

8.84 uuid_state Struct Reference

Collaboration diagram for uuid_state:



Data Fields

- uuid_time_t ts
- uuid_node_t node
- uint16_t cs

8.84.1 Detailed Description

Data type for UUID generator persistent state.

8.84.2 Field Documentation

8.84.2.1 cs

`uint16_t uuid_state::cs`

Saved clock sequence.

Referenced by read_state(), and write_state().

8.84.2.2 node

`uuid_node_t uuid_state::node`

Saved node ID.

Referenced by read_state(), and write_state().

8.84.2.3 ts

`uuid_time_t uuid_state::ts`

Saved timestamp.

Referenced by read_state(), and write_state().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/uuid.c

8.85 virtual_Dir_List Struct Reference

Collaboration diagram for virtual_Dir_List:

Data Fields

- struct virtual_Dir_List * **next**
- const void * **cookie**
- char **dirName** [NAME_SIZE]

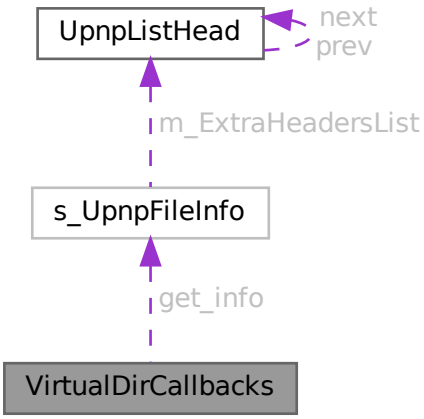
The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/VirtualDir.h

8.86 VirtualDirCallbacks Struct Reference

`#include <VirtualDir.h>`

Collaboration diagram for VirtualDirCallbacks:



Data Fields

- VDCallback_GetInfo get_info
- VDCallback_Open open
- VDCallback_Read read
- VDCallback_Write write
- VDCallback_Seek seek
- VDCallback_Close close

8.86.1 Detailed Description

The **VirtualDirCallbacks** structure contains the pointers to file-related callback functions a device application can register to virtualize URLs.

8.86.2 Field Documentation

8.86.2.1 close

`VDCallback_Close VirtualDirCallbacks::close`

Called by the web server to close a file opened via the **open** callback. It should return 0 on success, or a non-zero value on an error.

Referenced by http_RecvPostMessage(), http_SendMessage(), UpnpVirtualDir_set_CloseCallback(), and web_server_init().

8.86.2.2 get_info

`VDCallback_GetInfo VirtualDirCallbacks::get_info`

Called by the web server to query information on a file. The callback should return 0 on success or -1 on an error.

Referenced by process_request(), UpnpVirtualDir_set_GetInfoCallback(), and web_server_init().

8.86.2.3 open

`VDCallback_Open VirtualDirCallbacks::open`

Called by the web server to open a file. The callback should return a valid handle if the file can be opened. Otherwise, it should return `NULL` to signify an error.

Referenced by http_RecvPostMessage(), http_SendMessage(), UpnpVirtualDir_set_OpenCallback(), and web_server_init().

8.86.2.4 read

`VDCallback_Read VirtualDirCallbacks::read`

Called by the web server to perform a sequential read from an open file. The callback should copy **buflen** bytes from the file into the buffer.

Returns

An integer representing one of the following:

- `0`: The file contains no more data (EOF).
- $>$ `0`: A successful read of the number of bytes in the return code.
- $<$ `0`: An error occurred reading the file.

Referenced by http_SendMessage(), UpnpVirtualDir_set_ReadCallback(), and web_server_init().

8.86.2.5 seek

`VDCallback_Seek VirtualDirCallbacks::seek`

Called by the web server to move the file pointer, or offset, into an open file. The **origin** parameter determines where to start moving the file pointer. A value of `SEEK_CUR` moves the file pointer relative to where it is. The **offset** parameter can be either positive (move forward) or negative (move backward). `SEEK_END` moves relative to the end of the file. A positive **offset** extends the file. A negative **offset** moves backward in the file. Finally, `SEEK_SET` moves to an absolute position in the file. In this case, **offset** must be positive. The callback should return 0 on a successful seek or a non-zero value on an error.

Referenced by http_SendMessage(), UpnpVirtualDir_set_SeekCallback(), and web_server_init().

8.86.2.6 write

`VDCallback_Write VirtualDirCallbacks::write`

Called by the web server to perform a sequential write to an open file. The callback should write **buflen** bytes into the file from the buffer. It should return the actual number of bytes written, which might be less than **buflen** in the case of a write error.

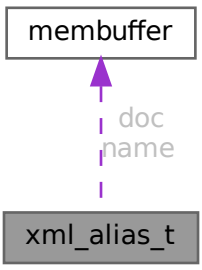
Referenced by http_RecvPostMessage(), UpnpVirtualDir_set_WriteCallback(), and web_server_init().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/VirtualDir.h

8.87 xml_alias_t Struct Reference

Collaboration diagram for xml_alias_t:



Data Fields

- membuffer name
- membuffer doc
- time_t last_modified
- int * ct

8.87.1 Field Documentation

8.87.1.1 ct

```
int* xml_alias_t::ct
```

Referenced by alias_grab(), alias_release(), glob_alias_init(), and web_server_set_alias().

8.87.1.2 doc

```
membuffer xml_alias_t::doc
```

the XML document contents

Referenced by alias_release(), get_alias(), glob_alias_init(), is_valid_alias(), web_server_callback(), and web_server_set_alias().

8.87.1.3 last_modified

```
time_t xml_alias_t::last_modified
```

Referenced by get_alias(), glob_alias_init(), and web_server_set_alias().

8.87.1.4 name

```
membuffer xml_alias_t::name
```

name of DOC from root; e.g.: /foo/bar/mydesc.xml

Referenced by alias_release(), get_alias(), glob_alias_init(), and web_server_set_alias().

The documentation for this struct was generated from the following file:

- /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/webserver.c

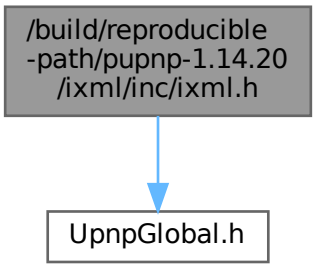
Chapter 9

File Documentation

9.1 /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h File Reference

```
#include "UpnpGlobal.h"
```

Include dependency graph for ixml.h:



This graph shows which files directly or indirectly include this file:

Data Structures

- struct _IXML_Node

  Data structure common to all types of nodes.
- struct _IXML_Document

  Data structure representing the DOM Document.

- struct _IXML_CDATASection

  Data structure representing a CDATA section node.
- struct _IXML_Element

  Data structure representing an Element node.
- struct _IXML_ATTR

  Data structure representing an Attribute node.
- struct _IXML_Text

  Data structure representing a Text node.
- struct _IXML_NodeList

  Data structure representing a list of nodes.
- struct _IXML_NamedNodeMap

  Data structure representing a list of named nodes.

Macros

- #define DOMString char *

  The type of DOM strings.

Functions

Interface Node

*The **Node** interface forms the primary datatype for all other DOM objects. Every other interface is derived from this interface, inheriting its functionality. For more information, refer to DOM2-Core page 34. (Note: within the IXML library the NamedNodeMap and NodeList interfaces are the only interfaces that are not DOM objects and hence do not inherit from Node)*

- UPNP_EXPORT_SPEC const char * ixmlNode_getNodeName (IXML_Node *nodeptr)

  *Returns the name of the **Node**, depending on what type of **Node** it is, in a read-only string.*
- UPNP_EXPORT_SPEC const char * ixmlNode_getNodeValue (IXML_Node *nodeptr)

  *Returns the value of the **Node** as a string.*
- UPNP_EXPORT_SPEC int ixmlNode_setNodeValue (IXML_Node *nodeptr, const char *newNodeValue)

  *Assigns a new value to a **Node**.*
- UPNP_EXPORT_SPEC unsigned short ixmlNode_getNodeType (IXML_Node *nodeptr)

  *Retrieves the type of a **Node**. Note that not all possible return values are actually implemented.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getParentNode (IXML_Node *nodeptr)

  *Retrieves the parent **Node** for a **Node**.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlNode_getChildNodes (IXML_Node *nodeptr)

  *Retrieves the list of children of a **Node** in a **NodeList** structure.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getFirstChild (IXML_Node *nodeptr)

  *Retrieves the first child **Node** of a **Node**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getLastChild (IXML_Node *nodeptr)

  *Retrieves the last child **Node** of a **Node**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getPreviousSibling (IXML_Node *nodeptr)

  *Retrieves the sibling **Node** immediately preceding this **Node**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_getNextSibling (IXML_Node *nodeptr)

  *Retrieves the sibling **Node** immediately following this **Node**.*
- UPNP_EXPORT_SPEC IXML_NamedNodeMap * ixmlNode_getAttributes (IXML_Node *nodeptr)

  *Retrieves the attributes of a **Node**, if it is an **Element** node, in a **NamedNodeMap** structure.*
- UPNP_EXPORT_SPEC IXML_Document * ixmlNode_getOwnerDocument (IXML_Node *nodeptr)

  *Retrieves the document object associated with this **Node**.*
- UPNP_EXPORT_SPEC const char * ixmlNode_getNamespaceURI (IXML_Node *nodeptr)

  *Retrieves the namespace URI for a **Node** as a **DOMString**.*
- UPNP_EXPORT_SPEC const char * ixmlNode_getPrefix (IXML_Node *nodeptr)

  Retrieves the namespace prefix, if present.

- UPNP_EXPORT_SPEC const char * ixmlNode_getLocalName (IXML_Node *nodeptr)
  *Retrieves the local name of a **Node**, if present.*
- UPNP_EXPORT_SPEC int ixmlNode_insertBefore (IXML_Node *nodeptr, IXML_Node *newChild, IXML_Node *refChild)
  *Inserts a new child **Node** before the existing child **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_replaceChild (IXML_Node *nodeptr, IXML_Node *newChild, IXML_Node *oldChild, IXML_Node **returnNode)
  *Replaces an existing child **Node** with a new child **Node** in the list of children of a **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_removeChild (IXML_Node *nodeptr, IXML_Node *oldChild, IXML_Node **returnNode)
  *Removes a child from the list of children of a **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_appendChild (IXML_Node *nodeptr, IXML_Node *newChild)
  *Appends a child **Node** to the list of children of a **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_hasChildNodes (IXML_Node *nodeptr)
  *Queries whether or not a **Node** has children.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNode_cloneNode (IXML_Node *nodeptr, int deep)
  *Clones a **Node**.*
- UPNP_EXPORT_SPEC int ixmlNode_hasAttributes (IXML_Node *nodeptr)
  *Queries whether this **Node** has attributes.*
- UPNP_EXPORT_SPEC void ixmlNode_free (IXML_Node *nodeptr)
  *Frees a **Node** and all **Nodes** in its subtree.*

Interface Attr

*The **Attr** interface represents an attribute of an **Element**. The document type definition (DTD) or schema usually dictate the allowable attributes and values for a particular element.*

For more information, refer to the Interface Attr section in the DOM2-Core.

- UPNP_EXPORT_SPEC void ixmlAttr_free (IXML_Attr *attrNode)
  *Frees an **Attr** node.*

Interface CDATASection

*The **CDATASection** is used to escape blocks of text containing characters that would otherwise be regarded as markup. CDATA sections cannot be nested. Their primary purpose is for including material such XML fragments, without needing to escape all the delimiters.*

For more information, refer to the Interface CDATASection section in the DOM2-Core.

- UPNP_EXPORT_SPEC void ixmlCDATASection_init (IXML_CDATASection *nodeptr)
  *Initializes a **CDATASection** node.*
- UPNP_EXPORT_SPEC void ixmlCDATASection_free (IXML_CDATASection *nodeptr)
  *Frees a **CDATASection** node.*

Interface Document

*The **Document** interface represents the entire XML document. In essence, it is the root of the document tree and provides the primary interface to the elements of the document.*

For more information, refer to the Interface Document section in the DOM2Core.

- UPNP_EXPORT_SPEC void ixmlDocument_init (IXML_Document *nodeptr)
  *Initializes a **Document** node.*
- UPNP_EXPORT_SPEC int ixmlDocument_createDocumentEx (IXML_Document **doc)
  *Creates a new empty **Document** node.*
- UPNP_EXPORT_SPEC IXML_Document * ixmlDocument_createDocument (void)
  *Creates a new empty **Document** node.*
- UPNP_EXPORT_SPEC int ixmlDocument_createElementEx (IXML_Document *doc, const char *tagName, IXML_Element **rtElement)
  *Creates a new **Element** node with the given tag name.*

- UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_createElement (IXML_Document *doc, const char *tagName)

  *Creates a new **Element** node with the given tag name.*
- UPNP_EXPORT_SPEC int ixmlDocument_createTextNodeEx (IXML_Document *doc, const char *data, IXML_Node **textNode)

  *Creates a new **Text** node with the given data.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlDocument_createTextNode (IXML_Document *doc, const char *data)

  *Creates a new **Text** node with the given data.*
- UPNP_EXPORT_SPEC int ixmlDocument_createCDATASectionEx (IXML_Document *doc, const char *data, IXML_CDATASection **cdNode)

  *Creates a new **CDATASection** node with given data.*
- UPNP_EXPORT_SPEC IXML_CDATASection * ixmlDocument_createCDATASection (IXML_Document *doc, const char *data)

  *Creates a new **CDATASection** node with given data.*
- UPNP_EXPORT_SPEC IXML_Attr * ixmlDocument_createAttribute (IXML_Document *doc, const char *name)

  *Creates a new **Attr** node with the given name.*
- UPNP_EXPORT_SPEC int ixmlDocument_createAttributeEx (IXML_Document *doc, const char *name, IXML_Attr **attrNode)

  *Creates a new **Attr** node with the given name.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlDocument_getElementsByTagName (IXML_Document *doc, const char *tagName)

  *Returns a **NodeList** of all **Elements** that match the given tag name in the order in which they were encountered in a preorder traversal of the **Document** tree.*
- UPNP_EXPORT_SPEC int ixmlDocument_createElementNSEx (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName, IXML_Element **rtElement)

  *Creates a new **Element** node in the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_createElementNS (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName)

  *Creates a new **Element** node in the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC int ixmlDocument_createAttributeNSEx (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName, IXML_Attr **attrNode)

  *Creates a new **Attr** node with the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC IXML_Attr * ixmlDocument_createAttributeNS (IXML_Document *doc, const char *namespaceURI, const char *qualifiedName)

  *Creates a new **Attribute** node with the given qualified name and namespace URI.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlDocument_getElementsByTagNameNS (IXML_Document *doc, const char *namespaceURI, const char *localName)

  *Returns a **NodeList** of **Elements** that match the given local name and namespace URI in the order they are encountered in a preorder traversal of the **Document** tree.*
- UPNP_EXPORT_SPEC IXML_Element * ixmlDocument_getElementById (IXML_Document *doc, const char *tagName)

  *Returns the **Element** whose `ID` matches that given id.*
- UPNP_EXPORT_SPEC void ixmlDocument_free (IXML_Document *doc)

  *Frees a **Document** object and all **Nodes** associated with it.*
- UPNP_EXPORT_SPEC int ixmlDocument_importNode (IXML_Document *doc, IXML_Node *importNode, int deep, IXML_Node **rtNode)

  *Imports a **Node** from another **Document** into this **Document**.*

Interface Element

*The **Element** interface represents an element in an XML document. Only **Elements** are allowed to have attributes, which are stored in the `attributes` member of a **Node**. The **Element** interface extends the **Node** interface and adds more operations to manipulate attributes.*

- UPNP_EXPORT_SPEC void ixmlElement_init (IXML_Element *element)

  *Initializes a **IXML_Element** node.*
- UPNP_EXPORT_SPEC const char * ixmlElement_getTagName (IXML_Element *element)

Returns the name of the tag as a constant string.

- UPNP_EXPORT_SPEC const char * ixmlElement_getAttribute (IXML_Element *element, const char *name)

  *Retrieves an attribute of an **Element** by name.*
- UPNP_EXPORT_SPEC int ixmlElement_setAttribute (IXML_Element *element, const char *name, const char *value)

  *Adds a new attribute to an **Element**.*
- UPNP_EXPORT_SPEC int ixmlElement_removeAttribute (IXML_Element *element, const char *name)

  Removes an attribute value by name. The attribute node is not removed.
- UPNP_EXPORT_SPEC IXML_Attr * ixmlElement_getAttributeNode (IXML_Element *element, const char *name)

  *Retrieves an attribute node by name. See **ixmlElement_getAttributeNodeNS** to retrieve an attribute node using a qualified name or namespace URI.*
- UPNP_EXPORT_SPEC int ixmlElement_setAttributeNode (IXML_Element *element, IXML_Attr *newAttr, IXML_Attr **rtAttr)

  *Adds a new attribute node to an **Element**.*
- UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNode (IXML_Element *element, IXML_Attr *oldAttr, IXML_Attr **rtAttr)

  *Removes the specified attribute node from an **Element**.*
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlElement_getElementsByTagName (IXML_Element *element, const char *tagName)

  *Returns a **NodeList** of all descendant **Elements** with a given tag name, in the order in which they are encountered in a pre-order traversal of this **Element** tree.*
- UPNP_EXPORT_SPEC const char * ixmlElement_getAttributeNS (IXML_Element *element, const char *namespaceURI, const char *localname)

  Retrieves an attribute value using the local name and namespace URI.
- UPNP_EXPORT_SPEC int ixmlElement_setAttributeNS (IXML_Element *element, const char *namespaceURI, const char *qualifiedName, const char *value)

  *Adds a new attribute to an **Element** using the local name and namespace URI.*
- UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNS (IXML_Element *element, const char *namespaceURI, const char *localName)

  Removes an attribute using the namespace URI and local name.
- UPNP_EXPORT_SPEC IXML_Attr * ixmlElement_getAttributeNodeNS (IXML_Element *element, const char *namespaceURI, const char *localName)

  *Retrieves an **Attr** node by local name and namespace URI.*
- UPNP_EXPORT_SPEC int ixmlElement_setAttributeNodeNS (IXML_Element *element, IXML_Attr *newAttr, IXML_Attr **rcAttr)

  Adds a new attribute node to the element node specified.
- UPNP_EXPORT_SPEC IXML_NodeList * ixmlElement_getElementsByTagNameNS (IXML_Element *element, const char *namespaceURI, const char *localName)

  *Returns a **NodeList** of all descendant **Elements** with a given local name and namespace in the order in which they are encountered in the pre-order traversal of the **Element** tree.*
- UPNP_EXPORT_SPEC int ixmlElement_hasAttribute (IXML_Element *element, const char *name)

  *Queries whether the **Element** has an attribute with the given name or a default value.*
- UPNP_EXPORT_SPEC int ixmlElement_hasAttributeNS (IXML_Element *element, const char *namespaceURI, const char *localName)

  *Queries whether the **Element** has an attribute with the given local name and namespace URI or has a default value for that attribute.*
- UPNP_EXPORT_SPEC void ixmlElement_free (IXML_Element *element)

  *Frees the given **Element** and any subtree of the **Element**.*

Interface NamedNodeMap

*A **NamedNodeMap** object represents a list of objects that can be accessed by name. A **NamedNodeMap** maintains the objects in no particular order. The **Node** interface uses a **NamedNodeMap** to maintain the attributes of a node.*

- UPNP_EXPORT_SPEC unsigned long ixmlNamedNodeMap_getLength (IXML_NamedNodeMap *nn↩ Map)

*Returns the number of items contained in this **NamedNodeMap**.*

- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_getNamedItem (IXML_NamedNodeMap *nnMap, const char *name)

  *Retrieves a **Node** from the **NamedNodeMap** by name.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_setNamedItem (IXML_NamedNodeMap *nnMap, IXML_Node *arg)

  *Adds a new **Node** to the **NamedNodeMap** using the **Node** name attribute.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_removeNamedItem (IXML_NamedNodeMap *nnMap, const char *name)

  *Removes a **Node** from a **NamedNodeMap** specified by name.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_item (IXML_NamedNodeMap *nnMap, unsigned long index)

  *Retrieves the indexth item in the map. If index is greater than or equal to the number of nodes in this map, this returns **NULL**.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_getNamedItemNS (IXML_NamedNodeMap *nnMap, const char **namespaceURI, const char *localName)

  *Retrieves a **Node** from a **NamedNodeMap** specified by namespace URI and local name.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_setNamedItemNS (IXML_NamedNodeMap *nnMap, IXML_Node *arg)

  *Adds a new **Node** to the **NamedNodeMap** using the **Node** local name and namespace URI attributes.*
- UPNP_EXPORT_SPEC IXML_Node * ixmlNamedNodeMap_removeNamedItemNS (IXML_NamedNodeMap *nnMap, const char *namespaceURI, const char *localName)

  *Removes a **Node** from a **NamedNodeMap** specified by namespace URI and local name.*
- UPNP_EXPORT_SPEC void ixmlNamedNodeMap_free (IXML_NamedNodeMap *nnMap)

  *Frees a **NamedNodeMap**.*

Interface NodeList

*The **NodeList** interface abstracts an ordered collection of nodes. Note that changes to the underlying nodes will change the nodes contained in a **NodeList**. The DOM2-Core refers to this as being live.*

- UPNP_EXPORT_SPEC IXML_Node * ixmlNodeList_item (IXML_NodeList *nList, unsigned long index)

  *Retrieves a **Node** from a **NodeList** specified by a numerical index.*
- UPNP_EXPORT_SPEC unsigned long ixmlNodeList_length (IXML_NodeList *nList)

  *Returns the number of **Nodes** in a **NodeList**.*
- UPNP_EXPORT_SPEC void ixmlNodeList_free (IXML_NodeList *nList)

  *Frees a **NodeList** object.*

IXML API

The IXML API contains utility functions that are not part of the standard DOM interfaces. They include functions to create a DOM structure from a file or buffer, create an XML file from a DOM structure, and manipulate DOMString objects.

- UPNP_EXPORT_SPEC char * ixmlPrintDocument (IXML_Document *doc)

  *Renders a **Node** and all sub-elements into an XML document representation.*
- UPNP_EXPORT_SPEC char * ixmlPrintNode (IXML_Node *doc)

  *Renders a **Node** and all sub-elements into an XML text representation.*
- UPNP_EXPORT_SPEC char * ixmlDocumenttoString (IXML_Document *doc)

  *Renders a **Node** and all sub-elements into an XML document representation.*
- UPNP_EXPORT_SPEC char * ixmlNodetoString (IXML_Node *doc)

  *Renders a **Node** and all sub-elements into an XML text representation. The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.*
- UPNP_EXPORT_SPEC void ixmlRelaxParser (char errorChar)

  Makes the XML parser more tolerant to malformed text.
- UPNP_EXPORT_SPEC IXML_Document * ixmlParseBuffer (const char *buffer)

  Parses an XML text buffer converting it into an IXML DOM representation.
- UPNP_EXPORT_SPEC int ixmlParseBufferEx (const char *buffer, IXML_Document **doc)

  Parses an XML text buffer converting it into an IXML DOM representation.

- UPNP_EXPORT_SPEC IXML_Document * ixmlLoadDocument (const char *xmlFile)
  Parses an XML text file converting it into an IXML DOM representation.
- UPNP_EXPORT_SPEC int ixmlLoadDocumentEx (const char *xmlFile, IXML_Document **doc)
  Parses an XML text file converting it into an IXML DOM representation.
- UPNP_EXPORT_SPEC char * ixmlCloneDOMString (const char *src)
  *Clones an existing **DOMString**.*
- UPNP_EXPORT_SPEC void ixmlFreeDOMString (char *buf)
  *Frees a **DOMString**.*

DOM Interfaces

The Document Object Model consists of a set of objects and interfaces for accessing and manipulating documents. IXML does not implement all the interfaces documented in the DOM2-Core recommendation but defines a subset of the most useful interfaces. A description of the supported interfaces and methods is presented in this section.

For a complete discussion on the object model, the object hierarchy, etc., refer to section 1.1 of the DOM2-Core recommendation.

- #define **DOCUMENTNODENAME** "#document"
- #define **TEXTNODENAME** "#text"
- #define **CDATANODENAME** "#cdata-section"
- enum IXML_NODE_TYPE {
  eINVALID_NODE = 0 , **eELEMENT_NODE** = 1 , **eATTRIBUTE_NODE** = 2 , **eTEXT_NODE** = 3 ,
  eCDATA_SECTION_NODE = 4 , **eENTITY_REFERENCE_NODE** = 5 , **eENTITY_NODE** = 6 , **e**↩
  PROCESSING_INSTRUCTION_NODE = 7 ,
  eCOMMENT_NODE = 8 , **eDOCUMENT_NODE** = 9 , **eDOCUMENT_TYPE_NODE** = 10 , **eDOCUMENT_**↩
  FRAGMENT_NODE = 11 ,
  eNOTATION_NODE = 12 }
  The type of the DOM node.
- enum IXML_ERRORCODE {
  IXML_SUCCESS = 0 , **IXML_INDEX_SIZE_ERR** = 1 , **IXML_DOMSTRING_SIZE_ERR** = 2 , **IXML_**↩
  HIERARCHY_REQUEST_ERR = 3 ,
  IXML_WRONG_DOCUMENT_ERR = 4 , **IXML_INVALID_CHARACTER_ERR** = 5 , **IXML_NO_DATA_**↩
  ALLOWED_ERR = 6 , **IXML_NO_MODIFICATION_ALLOWED_ERR** = 7 ,
  IXML_NOT_FOUND_ERR = 8 , **IXML_NOT_SUPPORTED_ERR** = 9 , **IXML_INUSE_ATTRIBUTE_ERR** =
  10 , **IXML_INVALID_STATE_ERR** = 11 ,
  IXML_SYNTAX_ERR = 12 , **IXML_INVALID_MODIFICATION_ERR** = 13 , **IXML_NAMESPACE_ERR** = 14 ,
  IXML_INVALID_ACCESS_ERR = 15 ,
  IXML_NO_SUCH_FILE = 101 , **IXML_INSUFFICIENT_MEMORY** = 102 , **IXML_FILE_DONE** = 104 , **IXML**↩
  _INVALID_PARAMETER = 105 ,
  IXML_FAILED = 106 , **IXML_INVALID_ITEM_NUMBER** = 107 }
  Error codes returned by the XML API, see the DOM spec.
- typedef struct _IXML_Document * **Docptr**
- typedef struct _IXML_Node * **Nodeptr**
- typedef struct _IXML_Node IXML_Node
  Data structure common to all types of nodes.
- typedef struct _IXML_Document IXML_Document
  Data structure representing the DOM Document.
- typedef struct _IXML_CDATASection IXML_CDATASection
  Data structure representing a CDATA section node.
- typedef struct _IXML_Element IXML_Element
  Data structure representing an Element node.
- typedef struct _IXML_ATTR IXML_Attr
  Data structure representing an Attribute node.

- typedef struct _IXML_Text IXML_Text

  Data structure representing a Text node.
- typedef struct _IXML_NodeList IXML_NodeList

  Data structure representing a list of nodes.
- typedef struct _IXML_NamedNodeMap IXML_NamedNodeMap

  Data structure representing a list of named nodes.

9.2 ixml.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef IXML_H
00033 #define IXML_H
00034
00043 #include "UpnpGlobal.h" /* For UPNP_EXPORT_SPEC */
00044
00048 #define DOMString char *
00049 /*typedef char *DOMString;*/
00050
00069 typedef enum
00070 {
00071         eINVALID_NODE = 0,
00072         eELEMENT_NODE = 1,
00073         eATTRIBUTE_NODE = 2,
00074         eTEXT_NODE = 3,
00075         eCDATA_SECTION_NODE = 4,
00076         eENTITY_REFERENCE_NODE = 5,
00077         eENTITY_NODE = 6,
00078         ePROCESSING_INSTRUCTION_NODE = 7,
00079         eCOMMENT_NODE = 8,
00080         eDOCUMENT_NODE = 9,
00081         eDOCUMENT_TYPE_NODE = 10,
00082         eDOCUMENT_FRAGMENT_NODE = 11,
00083         eNOTATION_NODE = 12
00084 } IXML_NODE_TYPE;
00085
00089 typedef enum
00090 {
00091         IXML_SUCCESS = 0,
00092
00093         IXML_INDEX_SIZE_ERR = 1,
00094         IXML_DOMSTRING_SIZE_ERR = 2,
00095         IXML_HIERARCHY_REQUEST_ERR = 3,
00096         IXML_WRONG_DOCUMENT_ERR = 4,
00097         IXML_INVALID_CHARACTER_ERR = 5,
00098         IXML_NO_DATA_ALLOWED_ERR = 6,
00099         IXML_NO_MODIFICATION_ALLOWED_ERR = 7,
```

```
00100         IXML_NOT_FOUND_ERR = 8,
00101         IXML_NOT_SUPPORTED_ERR = 9,
00102         IXML_INUSE_ATTRIBUTE_ERR = 10,
00103         IXML_INVALID_STATE_ERR = 11,
00104         IXML_SYNTAX_ERR = 12,
00105         IXML_INVALID_MODIFICATION_ERR = 13,
00106         IXML_NAMESPACE_ERR = 14,
00107         IXML_INVALID_ACCESS_ERR = 15,
00108
00109         IXML_NO_SUCH_FILE = 101,
00110         IXML_INSUFFICIENT_MEMORY = 102,
00111         IXML_FILE_DONE = 104,
00112         IXML_INVALID_PARAMETER = 105,
00113         IXML_FAILED = 106,
00114         IXML_INVALID_ITEM_NUMBER = 107
00115 } IXML_ERRORCODE;
00116
00117 #define DOCUMENTNODENAME "#document"
00118 #define TEXTNODENAME "#text"
00119 #define CDATANODENAME "#cdata-section"
00120
00121 typedef struct _IXML_Document *Docptr;
00122
00123 typedef struct _IXML_Node *Nodeptr;
00124
00125 #ifdef IXML_HAVE_SCRIPTSUPPORT
00129 typedef void (*IXML_BeforeFreeNode_t)(Nodeptr obj);
00130 #endif
00131
00135 typedef struct _IXML_Node
00136 {
00137         DOMString nodeName;
00138         DOMString nodeValue;
00139         IXML_NODE_TYPE nodeType;
00140         DOMString namespaceURI;
00141         DOMString prefix;
00142         DOMString localName;
00143         int readOnly;
00144
00145         Nodeptr parentNode;
00146         Nodeptr firstChild;
00147         Nodeptr prevSibling;
00148         Nodeptr nextSibling;
00149         Nodeptr firstAttr;
00150         Docptr ownerDocument;
00151 #ifdef IXML_HAVE_SCRIPTSUPPORT
00152         void *ctag; /* custom tag */
00153 #endif
00154 } IXML_Node;
00155
00159 typedef struct _IXML_Document
00160 {
00161         IXML_Node n;
00162 } IXML_Document;
00163
00167 typedef struct _IXML_CDATASection
00168 {
00169         IXML_Node n;
00170 } IXML_CDATASection;
00171
00175 typedef struct _IXML_Element
00176 {
00177         IXML_Node n;
00178         DOMString tagName;
00179 } IXML_Element;
00180
00184 typedef struct _IXML_ATTR
00185 {
00186         IXML_Node n;
00187         int specified;
00188         IXML_Element *ownerElement;
00189 } IXML_Attr;
00190
00194 typedef struct _IXML_Text
00195 {
00196         IXML_Node n;
00197 } IXML_Text;
00198
00202 typedef struct _IXML_NodeList
00203 {
00204         IXML_Node *nodeItem;
00205         struct _IXML_NodeList *next;
00206 } IXML_NodeList;
00207
00211 typedef struct _IXML_NamedNodeMap
00212 {
00213         IXML_Node *nodeItem;
```

```
00214         struct _IXML_NamedNodeMap *next;
00215 } IXML_NamedNodeMap;
00216
00217 /* @} DOM Interfaces */
00218
00219 #ifdef __cplusplus
00220 extern "C" {
00221 #endif
00222
00245 UPNP_EXPORT_SPEC const DOMString ixmlNode_getNodeName(
00247         IXML_Node *nodeptr);
00248
00257 UPNP_EXPORT_SPEC const DOMString ixmlNode_getNodeValue(
00259         IXML_Node *nodeptr);
00260
00273 UPNP_EXPORT_SPEC int ixmlNode_setNodeValue(
00275         IXML_Node *nodeptr,
00277         const char *newNodeValue);
00278
00285 UPNP_EXPORT_SPEC unsigned short ixmlNode_getNodeType(
00287         IXML_Node *nodeptr);
00288
00295 UPNP_EXPORT_SPEC IXML_Node *ixmlNode_getParentNode(
00297         IXML_Node *nodeptr);
00298
00308 UPNP_EXPORT_SPEC IXML_NodeList *ixmlNode_getChildNodes(
00310         IXML_Node *nodeptr);
00311
00318 UPNP_EXPORT_SPEC IXML_Node *ixmlNode_getFirstChild(
00320         IXML_Node *nodeptr);
00321
00328 UPNP_EXPORT_SPEC IXML_Node *ixmlNode_getLastChild(
00330         IXML_Node *nodeptr);
00331
00338 UPNP_EXPORT_SPEC IXML_Node *ixmlNode_getPreviousSibling(
00340         IXML_Node *nodeptr);
00341
00348 UPNP_EXPORT_SPEC IXML_Node *ixmlNode_getNextSibling(
00350         IXML_Node *nodeptr);
00351
00358 UPNP_EXPORT_SPEC IXML_NamedNodeMap *ixmlNode_getAttributes(
00360         IXML_Node *nodeptr);
00361
00372 UPNP_EXPORT_SPEC IXML_Document *ixmlNode_getOwnerDocument(
00374         IXML_Node *nodeptr);
00375
00385 UPNP_EXPORT_SPEC const DOMString ixmlNode_getNamespaceURI(
00387         IXML_Node *nodeptr);
00388
00399 UPNP_EXPORT_SPEC const DOMString ixmlNode_getPrefix(
00401         IXML_Node *nodeptr);
00402
00414 UPNP_EXPORT_SPEC const DOMString ixmlNode_getLocalName(
00416         IXML_Node *nodeptr);
00417
00443 UPNP_EXPORT_SPEC int ixmlNode_insertBefore(
00446         IXML_Node *nodeptr,
00448         IXML_Node *newChild,
00451         IXML_Node *refChild);
00452
00475 UPNP_EXPORT_SPEC int ixmlNode_replaceChild(
00478         IXML_Node *nodeptr,
00480         IXML_Node *newChild,
00482         IXML_Node *oldChild,
00485         IXML_Node **returnNode);
00486
00503 UPNP_EXPORT_SPEC int ixmlNode_removeChild(
00505         IXML_Node *nodeptr,
00507         IXML_Node *oldChild,
00510         IXML_Node **returnNode);
00511
00529 UPNP_EXPORT_SPEC int ixmlNode_appendChild(
00531         IXML_Node *nodeptr,
00533         IXML_Node *newChild);
00534
00540 UPNP_EXPORT_SPEC int ixmlNode_hasChildNodes(
00542         IXML_Node *nodeptr);
00543
00555 UPNP_EXPORT_SPEC IXML_Node *ixmlNode_cloneNode(
00557         IXML_Node *nodeptr,
00560         int deep);
00561
00569 UPNP_EXPORT_SPEC int ixmlNode_hasAttributes(
00571         IXML_Node *nodeptr);
00572
00576 UPNP_EXPORT_SPEC void ixmlNode_free(
00581         IXML_Node *nodeptr);
```

```
00582
00583 #ifdef IXML_HAVE_SCRIPTSUPPORT
00587 UPNP_EXPORT_SPEC void ixmlNode_setCTag(
00589         IXML_Node *nodeptr,
00591         void *ctag);
00592
00596 UPNP_EXPORT_SPEC void *ixmlNode_getCTag(
00598         IXML_Node *nodeptr);
00599 #endif
00600 /* @} Interface Node */
00601
00618 UPNP_EXPORT_SPEC void ixmlAttr_free(
00620         IXML_Attr *attrNode);
00621
00622 /* @} Interface Attr */
00623
00641 UPNP_EXPORT_SPEC void ixmlCDATASection_init(
00643         IXML_CDATASection *nodeptr);
00644
00648 UPNP_EXPORT_SPEC void ixmlCDATASection_free(
00650         IXML_CDATASection *nodeptr);
00651
00652 /* @} Interface CDATASection */
00653
00670 UPNP_EXPORT_SPEC void ixmlDocument_init(
00672         IXML_Document *nodeptr);
00673
00686 UPNP_EXPORT_SPEC int ixmlDocument_createDocumentEx(
00689         IXML_Document **doc);
00690
00697 UPNP_EXPORT_SPEC IXML_Document *ixmlDocument_createDocument(void);
00698
00717 UPNP_EXPORT_SPEC int ixmlDocument_createElementEx(
00719         IXML_Document *doc,
00721         const DOMString tagName,
00724         IXML_Element **rtElement);
00725
00737 UPNP_EXPORT_SPEC IXML_Element *ixmlDocument_createElement(
00739         IXML_Document *doc,
00741         const DOMString tagName);
00742
00757 UPNP_EXPORT_SPEC int ixmlDocument_createTextNodeEx(
00759         IXML_Document *doc,
00762         const DOMString data,
00764         IXML_Node **textNode);
00765
00771 UPNP_EXPORT_SPEC IXML_Node *ixmlDocument_createTextNode(
00773         IXML_Document *doc,
00776         const DOMString data);
00777
00792 UPNP_EXPORT_SPEC int ixmlDocument_createCDATASectionEx(
00794         IXML_Document *doc,
00796         const DOMString data,
00798         IXML_CDATASection **cdNode);
00799
00805 UPNP_EXPORT_SPEC IXML_CDATASection *ixmlDocument_createCDATASection(
00807         IXML_Document *doc,
00809         const DOMString data);
00810
00820 UPNP_EXPORT_SPEC IXML_Attr *ixmlDocument_createAttribute(
00822         IXML_Document *doc,
00824         const DOMString name);
00825
00840 UPNP_EXPORT_SPEC int ixmlDocument_createAttributeEx(
00842         IXML_Document *doc,
00844         const DOMString name,
00846         IXML_Attr **attrNode);
00847
00856 UPNP_EXPORT_SPEC IXML_NodeList *ixmlDocument_getElementsByTagName(
00858         IXML_Document *doc,
00860         const DOMString tagName);
00861
00862 /*
00863  * introduced in DOM level 2
00864  */
00865
00881 UPNP_EXPORT_SPEC int ixmlDocument_createElementNSEx(
00883         IXML_Document *doc,
00885         const DOMString namespaceURI,
00887         const DOMString qualifiedName,
00890         IXML_Element **rtElement);
00891
00900 UPNP_EXPORT_SPEC IXML_Element *ixmlDocument_createElementNS(
00902         IXML_Document *doc,
00904         const DOMString namespaceURI,
00906         const DOMString qualifiedName);
00907
```

```
00923 UPNP_EXPORT_SPEC int ixmlDocument_createAttributeNSEx(
00925         IXML_Document *doc,
00927         const DOMString namespaceURI,
00929         const DOMString qualifiedName,
00932         IXML_Attr **attrNode);
00933
00942 UPNP_EXPORT_SPEC IXML_Attr *ixmlDocument_createAttributeNS(
00944         IXML_Document *doc,
00946         const DOMString namespaceURI,
00948         const DOMString qualifiedName);
00949
00961 UPNP_EXPORT_SPEC IXML_NodeList *ixmlDocument_getElementsByTagNameNS(
00963         IXML_Document *doc,
00966         const DOMString namespaceURI,
00969         const DOMString localName);
00970
00976 UPNP_EXPORT_SPEC IXML_Element *ixmlDocument_getElementById(
00978         IXML_Document *doc,
00980         const DOMString tagName);
00981
00989 UPNP_EXPORT_SPEC void ixmlDocument_free(
00991         IXML_Document *doc);
00992
01023 UPNP_EXPORT_SPEC int ixmlDocument_importNode(
01025         IXML_Document *doc,
01027         IXML_Node *importNode,
01030         int deep,
01032         IXML_Node **rtNode);
01033
01034 /* @} Interface Document */
01035
01051 UPNP_EXPORT_SPEC void ixmlElement_init(
01053         IXML_Element *element);
01054
01060 UPNP_EXPORT_SPEC const DOMString ixmlElement_getTagName(
01062         IXML_Element *element);
01063
01070 UPNP_EXPORT_SPEC const DOMString ixmlElement_getAttribute(
01072         IXML_Element *element,
01074         const DOMString name);
01075
01092 UPNP_EXPORT_SPEC int ixmlElement_setAttribute(
01094         IXML_Element *element,
01096         const DOMString name,
01099         const DOMString value);
01100
01109 UPNP_EXPORT_SPEC int ixmlElement_removeAttribute(
01111         IXML_Element *element,
01113         const DOMString name);
01114
01123 UPNP_EXPORT_SPEC IXML_Attr *ixmlElement_getAttributeNode(
01125         IXML_Element *element,
01127         const DOMString name);
01128
01146 UPNP_EXPORT_SPEC int ixmlElement_setAttributeNode(
01148         IXML_Element *element,
01150         IXML_Attr *newAttr,
01153         IXML_Attr **rtAttr);
01154
01165 UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNode(
01167         IXML_Element *element,
01169         IXML_Attr *oldAttr,
01172         IXML_Attr **rtAttr);
01173
01181 UPNP_EXPORT_SPEC IXML_NodeList *ixmlElement_getElementsByTagName(
01183         IXML_Element *element,
01185         const DOMString tagName);
01186
01187 /*
01188  * Introduced in DOM 2
01189  */
01190
01197 UPNP_EXPORT_SPEC const DOMString ixmlElement_getAttributeNS(
01199         IXML_Element *element,
01201         const DOMString namespaceURI,
01203         const DOMString localname);
01204
01227 UPNP_EXPORT_SPEC int ixmlElement_setAttributeNS(
01229         IXML_Element *element,
01231         const DOMString namespaceURI,
01233         const DOMString qualifiedName,
01235         const DOMString value);
01236
01248 UPNP_EXPORT_SPEC int ixmlElement_removeAttributeNS(
01250         IXML_Element *element,
01252         const DOMString namespaceURI,
01254         const DOMString localName);
```

```
01255
01262 UPNP_EXPORT_SPEC IXML_Attr *ixmlElement_getAttributeNodeNS(
01264         IXML_Element *element,
01266         const DOMString namespaceURI,
01268         const DOMString localName);
01269
01290 UPNP_EXPORT_SPEC int ixmlElement_setAttributeNodeNS(
01292         IXML_Element *element,
01294         IXML_Attr *newAttr,
01296         IXML_Attr **rcAttr);
01297
01305 UPNP_EXPORT_SPEC IXML_NodeList *ixmlElement_getElementsByTagNameNS(
01307         IXML_Element *element,
01311         const DOMString namespaceURI,
01314         const DOMString localName);
01315
01323 UPNP_EXPORT_SPEC int ixmlElement_hasAttribute(
01325         IXML_Element *element,
01327         const DOMString name);
01328
01336 UPNP_EXPORT_SPEC int ixmlElement_hasAttributeNS(
01338         IXML_Element *element,
01340         const DOMString namespaceURI,
01342         const DOMString localName);
01343
01347 UPNP_EXPORT_SPEC void ixmlElement_free(
01349         IXML_Element *element);
01350
01351 /* @} Interface Element */
01352
01369 UPNP_EXPORT_SPEC unsigned long ixmlNamedNodeMap_getLength(
01371         IXML_NamedNodeMap *nnMap);
01372
01379 UPNP_EXPORT_SPEC IXML_Node *ixmlNamedNodeMap_getNamedItem(
01381         IXML_NamedNodeMap *nnMap,
01383         const DOMString name);
01384
01392 UPNP_EXPORT_SPEC IXML_Node *ixmlNamedNodeMap_setNamedItem(
01394         IXML_NamedNodeMap *nnMap,
01396         IXML_Node *arg);
01397
01403 UPNP_EXPORT_SPEC IXML_Node *ixmlNamedNodeMap_removeNamedItem(
01405         IXML_NamedNodeMap *nnMap,
01407         const DOMString name);
01408
01416 UPNP_EXPORT_SPEC IXML_Node *ixmlNamedNodeMap_item(
01418         IXML_NamedNodeMap *nnMap,
01420         unsigned long index);
01421
01422 /*
01423  * Introduced in DOM level 2
01424  */
01425
01432 UPNP_EXPORT_SPEC IXML_Node *ixmlNamedNodeMap_getNamedItemNS(
01434         IXML_NamedNodeMap *nnMap,
01436         const DOMString *namespaceURI,
01438         const DOMString localName);
01439
01447 UPNP_EXPORT_SPEC IXML_Node *ixmlNamedNodeMap_setNamedItemNS(
01449         IXML_NamedNodeMap *nnMap,
01451         IXML_Node *arg);
01452
01459 UPNP_EXPORT_SPEC IXML_Node *ixmlNamedNodeMap_removeNamedItemNS(
01461         IXML_NamedNodeMap *nnMap,
01463         const DOMString namespaceURI,
01465         const DOMString localName);
01466
01471 UPNP_EXPORT_SPEC void ixmlNamedNodeMap_free(
01473         IXML_NamedNodeMap *nnMap);
01474
01475 /* @} Interface NodeMap */
01476
01494 UPNP_EXPORT_SPEC IXML_Node *ixmlNodeList_item(
01496         IXML_NodeList *nList,
01498         unsigned long index);
01499
01505 UPNP_EXPORT_SPEC unsigned long ixmlNodeList_length(
01508         IXML_NodeList *nList);
01509
01516 UPNP_EXPORT_SPEC void ixmlNodeList_free(
01518         IXML_NodeList *nList);
01519
01520 /* @} Interface NodeList */
01521
01554 UPNP_EXPORT_SPEC DOMString ixmlPrintDocument(
01556         IXML_Document *doc);
01557
```

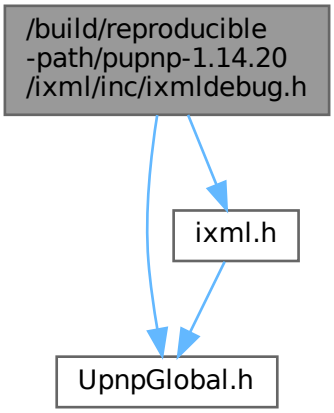
```
01575 UPNP_EXPORT_SPEC DOMString ixmlPrintNode(
01577         IXML_Node *doc);
01578
01597 UPNP_EXPORT_SPEC DOMString ixmlDocumenttoString(
01599         IXML_Document *doc);
01600
01615 UPNP_EXPORT_SPEC DOMString ixmlNodetoString(
01617         IXML_Node *doc);
01618
01622 UPNP_EXPORT_SPEC void ixmlRelaxParser(
01631         char errorChar);
01632
01633 #ifdef IXML_HAVE_SCRIPTSUPPORT
01637 UPNP_EXPORT_SPEC void ixmlSetBeforeFree(
01643         IXML_BeforeFreeNode_t hndlr);
01644 #endif
01645
01652 UPNP_EXPORT_SPEC IXML_Document *ixmlParseBuffer(
01655         const char *buffer);
01656
01672 UPNP_EXPORT_SPEC int ixmlParseBufferEx(
01675         const char *buffer,
01678         IXML_Document **doc);
01679
01685 UPNP_EXPORT_SPEC IXML_Document *ixmlLoadDocument(
01687         const char *xmlFile);
01688
01703 UPNP_EXPORT_SPEC int ixmlLoadDocumentEx(
01705         const char *xmlFile,
01708         IXML_Document **doc);
01709
01716 UPNP_EXPORT_SPEC DOMString ixmlCloneDOMString(
01718         const DOMString src);
01719
01723 UPNP_EXPORT_SPEC void ixmlFreeDOMString(
01725         DOMString buf);
01726
01727 /* @} IXML API */
01728
01729 #ifdef __cplusplus
01730 }
01731 #endif
01732
01733 /* @} XMLAPI XML API */
01734
01735 #endif /* IXML_H */
```

9.3 /build/reproducible-path/pupnp-1.14.20/ixml/inc/ixmldebug.h File Reference

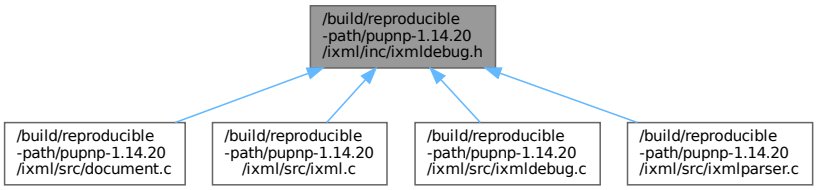
Auxiliar routines to aid debugging.

```
#include "UpnpGlobal.h"
#include "ixml.h"
```

Include dependency graph for ixmldebug.h:



This graph shows which files directly or indirectly include this file:



Functions

- void IxmlPrintf (const char *DbgFileName, int DbgLineNo, const char *FunctionName, const char *FmtStr,...)

  Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.
- void printNodes (IXML_Node *tmpRoot, int depth)

  Print the node names and values of a XML tree.

9.3.1 Detailed Description

Auxiliar routines to aid debugging.

9.3.2 Function Documentation

9.3.2.1 IxmlPrintf()

```
void IxmlPrintf (
            const char * DbgFileName,
            int DbgLineNo,
            const char * FunctionName,
            const char * FmtStr,
             ... )
```

Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.

Parameters

in	*DbgFileName*	The file name, usually **FILE**.
in	*DbgLineNo*	The line number, usually **LINE** or a variable that got the **LINE** at the appropriate place.
in	*FunctionName*	The function name.
in	*FmtStr*	Printf like format specification.
in	...	Printf like Variable number of arguments that will go in the debug statement.

Referenced by ixmlDomTreetoString(), ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), Parser_copyToken(), Parser_getChar(), Parser_getNextNode(), Parser_processAttribute(), Parser_processContent(), Parser_processETag(), Parser_UTF8ToInt(), Parser_xmlNamespace(), and printNodes().

9.3.2.2 printNodes()

```
void printNodes (
            IXML_Node * tmpRoot,
            int depth )
```

Print the node names and values of a XML tree.

Parameters

in	*tmpRoot*	The root of the tree to print.
in	*depth*	The depth to print.

References DOMString, ixmlNode_getChildNodes(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), ixmlNode_getNodeValue(), ixmlNodeList_item(), IxmlPrintf(), and printNodes().

Referenced by printNodes().

9.4 ixmldebug.h

Go to the documentation of this file.

```
00001
00002
00003 #ifndef IXMLDEBUG_H
```

```
00004 #define IXMLDEBUG_H
00005
00006 #include "UpnpGlobal.h"
00007 #include "ixml.h"
00008
00019 #ifdef DEBUG
00020 void IxmlPrintf(
00022         const char *DbgFileName,
00025         int DbgLineNo,
00027         const char *FunctionName,
00029         const char *FmtStr,
00032         ...)
00033         #if (__GNUC__ >= 3)
00034         /* This enables printf like format checking by the compiler */
00035         __attribute__((format(__printf__, 4, 5)))
00036         #endif
00037         ;
00038 #else  /* DEBUG */
00039 static UPNP_INLINE void IxmlPrintf(const char *FmtStr, ...) { (void)FmtStr; }
00040 #endif /* DEBUG */
00041
00045 #ifdef DEBUG
00046 void printNodes(
00048         IXML_Node *tmpRoot,
00050         int depth);
00051 #else
00052 static UPNP_INLINE void printNodes(IXML_Node *tmpRoot, int depth)
00053 {
00054         (void)tmpRoot;
00055         (void)depth;
00056 }
00057 #endif
00058
00059 #endif /* IXMLDEBUG_H */
```

9.5 posix_overwrites.h

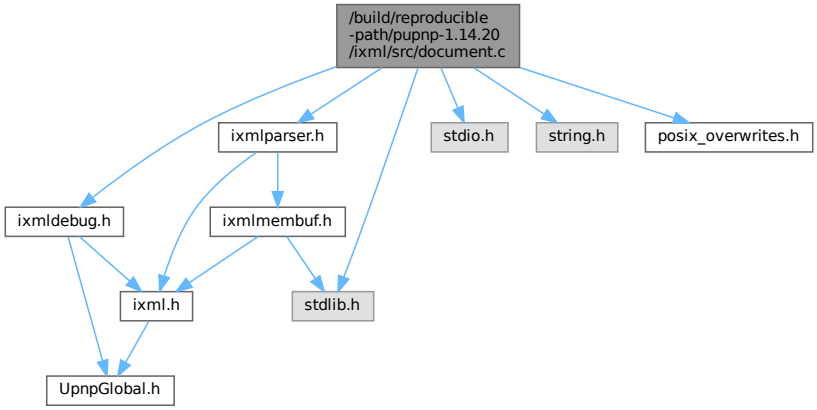
```
00001 #ifndef POSIX_OVERWRITES_H
00002 #define POSIX_OVERWRITES_H
00003
00004 #ifdef _WIN32
00005
00006         /* POSIX names for functions */
00007         #define fileno _fileno
00008         #define unlink _unlink
00009         #define strcasecmp _stricmp
00010         #define strdup _strdup
00011         #define stricmp _stricmp
00012         #define strncasecmp strnicmp
00013         #define strnicmp _strnicmp
00014
00015         /* Secure versions of functions */
00016         /* Explicitly disable warnings by pragma/define, see:
00017          * https://www.codegrepper.com/code-examples/c/crt+secure+no+warnings */
00018         #pragma warning(disable : 4996)
00019         #define _CRT_SECURE_NO_WARNINGS
00020         #if 0
00021                 /*
00022                  * The current issues with those 4 defines:
00023                  * - strncpy redefinition is wrong
00024                  * - Theses functions assume they are being called on C arrays
00025                  * only. Using `countof` on a heap allocated pointer is
00026                  * undefined behavior and `sizeof` will only return the byte
00027                  * size of the pointer.
00028                  *
00029                  * The reason we can't pin-point the places where it fails is
00030                  * because *_s functions have a significantly different
00031                  * behaviour than the replaced functions and have actual error
00032                  * returns values that are simply ignored here, leading to
00033                  * numerous unseen regressions.
00034                  *
00035                  * A first step could be to actually crash or log on _s failures
00036                  * to detect the potentials overflows or bad usages of the
00037                  * wrappers.
00038                  */
00039                 #define strcat(arg1, arg2) strcat_s(arg1, sizeof(arg1), arg2)
00040                 #define strcpy(arg1, arg2) strcpy_s(arg1, _countof(arg1), arg2)
00041                 #define strncpy(arg1, arg2, arg3) \
00042                         strncpy_s(arg1, arg3, arg2, arg3)
00043                 #define sprintf(arg1, ...) \
00044                         sprintf_s(arg1, sizeof(arg1), __VA_ARGS__)
00045         #endif
00046
```

```
00047 #endif /* _WIN32 */
00048
00049 #endif /* POSIX_OVERWRITES_H */
```

9.6 /build/reproducible-path/pupnp-1.14.20/ixml/src/document.c File Reference

```
#include "ixmldebug.h"
#include "ixmlparser.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for document.c:



Functions

- void ixmlDocument_init (IXML_Document *doc)

  *Initializes a **Document** node.*
- void ixmlDocument_free (IXML_Document *doc)

  *Frees a **Document** object and all **Nodes** associated with it.*
- static void ixmlDocument_setOwnerDocument (IXML_Document *doc, IXML_Node *nodeptr)
- int ixmlDocument_importNode (IXML_Document *doc, IXML_Node *importNode, int deep, IXML_Node **rtNode)

  *Imports a **Node** from another **Document** into this **Document**.*
- int **ixmlDocument_createElementEx** (IXML_Document *doc, const DOMString tagName, IXML_Element **rtElement)
- IXML_Element * **ixmlDocument_createElement** (IXML_Document *doc, const DOMString tagName)
- int ixmlDocument_createDocumentEx (IXML_Document **rtDoc)

  *Creates a new empty **Document** node.*
- IXML_Document * ixmlDocument_createDocument ()

  *Creates a new empty **Document** node.*
- int **ixmlDocument_createTextNodeEx** (IXML_Document *doc, const DOMString data, IXML_Node **text↩ Node)

- IXML_Node * **ixmlDocument_createTextNode** (IXML_Document *doc, const DOMString data)
- int **ixmlDocument_createAttributeEx** (IXML_Document *doc, const DOMString name, IXML_Attr **rtAttr)
- IXML_Attr * **ixmlDocument_createAttribute** (IXML_Document *doc, const DOMString name)
- int **ixmlDocument_createAttributeNSEx** (IXML_Document *doc, const DOMString namespaceURI, const DOMString qualifiedName, IXML_Attr **rtAttr)
- IXML_Attr * **ixmlDocument_createAttributeNS** (IXML_Document *doc, const DOMString namespaceURI, const DOMString qualifiedName)
- int **ixmlDocument_createCDATASectionEx** (IXML_Document *doc, const DOMString data, IXML_CDATASection **rtCD)
- IXML_CDATASection * **ixmlDocument_createCDATASection** (IXML_Document *doc, const DOMString data)
- int **ixmlDocument_createElementNSEx** (IXML_Document *doc, const DOMString namespaceURI, const DOMString qualifiedName, IXML_Element **rtElement)
- IXML_Element * **ixmlDocument_createElementNS** (IXML_Document *doc, const DOMString namespaceURI, const DOMString qualifiedName)
- IXML_NodeList * **ixmlDocument_getElementsByTagName** (IXML_Document *doc, const DOMString tagName)
- IXML_NodeList * **ixmlDocument_getElementsByTagNameNS** (IXML_Document *doc, const DOMString namespaceURI, const DOMString localName)
- IXML_Element * **ixmlDocument_getElementById** (IXML_Document *doc, const DOMString tagName)

9.6.1 Function Documentation

9.6.1.1 ixmlDocument_setOwnerDocument()

```
static void ixmlDocument_setOwnerDocument (
            IXML_Document * doc,
            IXML_Node * nodeptr )  [static]
```

When this function is called first time, nodeptr is the root of the subtree, so it is not necessay to do two steps recursion.

Internal function called by ixmlDocument_importNode

Parameters

in	*doc*	The document node.
in	*nodeptr*	

Todo documentation.

References ixmlDocument_setOwnerDocument(), ixmlNode_getFirstChild(), and ixmlNode_getNextSibling().

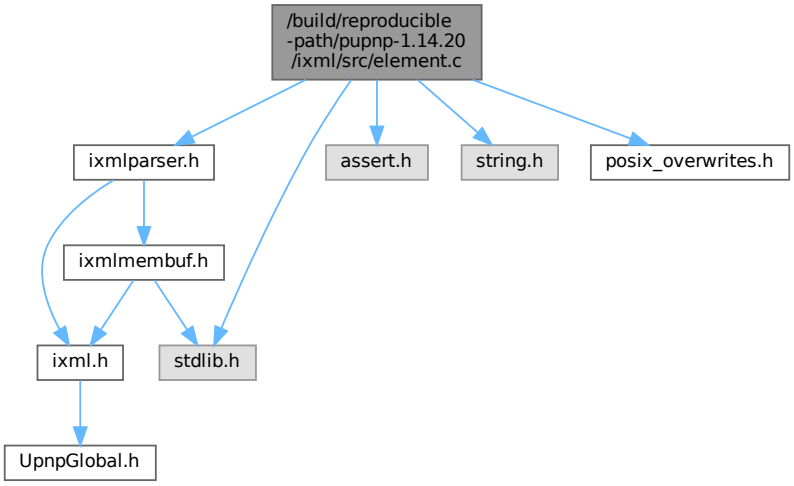
Referenced by ixmlDocument_importNode(), and ixmlDocument_setOwnerDocument().

9.7 /build/reproducible-path/pupnp-1.14.20/ixml/src/element.c File Reference

```
#include "ixmlparser.h"
#include <assert.h>
```

```
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for element.c:



Functions

- void ixmlElement_init (IXML_Element *element)

  *Initializes a **IXML_Element** node.*
- const DOMString ixmlElement_getTagName (IXML_Element *element)

  Returns the name of the tag as a constant string.
- int ixmlElement_setTagName (IXML_Element *element, const char *tagName)

  Set the given element's tagName.
- const DOMString **ixmlElement_getAttribute** (IXML_Element *element, const DOMString name)
- int **ixmlElement_setAttribute** (IXML_Element *element, const DOMString name, const DOMString value)
- int **ixmlElement_removeAttribute** (IXML_Element *element, const DOMString name)
- IXML_Attr * **ixmlElement_getAttributeNode** (IXML_Element *element, const DOMString name)
- int ixmlElement_setAttributeNode (IXML_Element *element, IXML_Attr *newAttr, IXML_Attr **rtAttr)

  *Adds a new attribute node to an **Element**.*
- static IXML_Node * ixmlElement_findAttributeNode (IXML_Element *element, IXML_Attr *oldAttr)

  Find a attribute node whose contents are the same as the oldAttr.
- int ixmlElement_removeAttributeNode (IXML_Element *element, IXML_Attr *oldAttr, IXML_Attr **rtAttr)

  *Removes the specified attribute node from an **Element**.*
- IXML_NodeList * **ixmlElement_getElementsByTagName** (IXML_Element *element, const DOMString tagName)
- const DOMString **ixmlElement_getAttributeNS** (IXML_Element *element, const DOMString namespace↩ URI, const DOMString localName)
- int **ixmlElement_setAttributeNS** (IXML_Element *element, const DOMString namespaceURI, const DOMString qualifiedName, const DOMString value)
- int **ixmlElement_removeAttributeNS** (IXML_Element *element, const DOMString namespaceURI, const DOMString localName)
- IXML_Attr * **ixmlElement_getAttributeNodeNS** (IXML_Element *element, const DOMString namespace↩ URI, const DOMString localName)

- int ixmlElement_setAttributeNodeNS (IXML_Element *element, IXML_Attr *newAttr, IXML_Attr **rtAttr)

  Adds a new attribute node to the element node specified.
- IXML_NodeList * **ixmlElement_getElementsByTagNameNS** (IXML_Element *element, const DOMString namespaceURI, const DOMString localName)
- int **ixmlElement_hasAttribute** (IXML_Element *element, const DOMString name)
- int **ixmlElement_hasAttributeNS** (IXML_Element *element, const DOMString namespaceURI, const DOMString localName)
- void ixmlElement_free (IXML_Element *element)

  *Frees the given **Element** and any subtree of the **Element**.*

9.7.1 Function Documentation

9.7.1.1 ixmlElement_findAttributeNode()

```
static IXML_Node * ixmlElement_findAttributeNode (
            IXML_Element * element,
            IXML_Attr * oldAttr )  [static]
```

Find a attribute node whose contents are the same as the oldAttr.

Returns

If found, the attribute node is returned, otherwise **NULL** is returned.

Parameters

in	*element*	The element to search for the attribute.
in	*oldAttr*	The attribute node to match.

References ixmlNode_compare().

Referenced by ixmlElement_removeAttributeNode().

9.7.1.2 ixmlElement_setTagName()

```
int ixmlElement_setTagName (
            IXML_Element * element,
            const char * tagName )
```

Set the given element's tagName.

Returns

One of the following:

- **IXML_SUCCESS**, if successfull.
- **IXML_FAILED**, if element of tagname is **NULL**.
- **IXML_INSUFFICIENT_MEMORY**, if there is no memory to allocate the buffer for the element's tagname.

Parameters

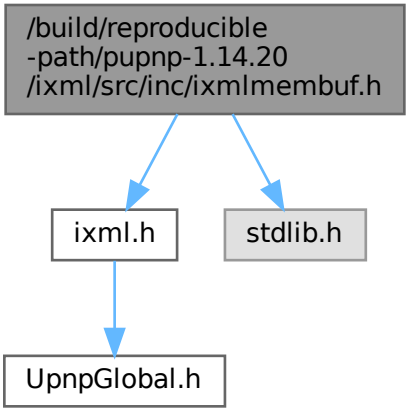
in	*element*	The element to change the tagname.
in	*tagName*	The new tagName for the element.

Referenced by ixmlNode_cloneElement().

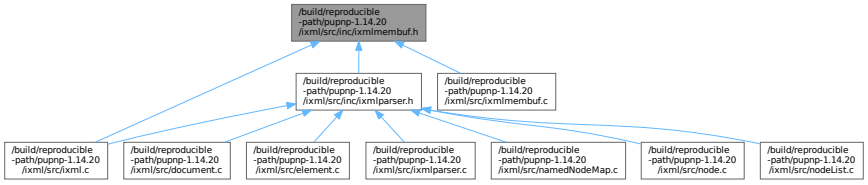
9.8 /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlmembuf.h File Reference

```
#include "ixml.h"
#include <stdlib.h>
```

Include dependency graph for ixmlmembuf.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct ixml_membuf

  The ixml_membuf type.

Macros

- #define **MINVAL**(a, b) ((a) < (b) ? (a) : (b))
- #define **MAXVAL**(a, b) ((a) > (b) ? (a) : (b))
- #define **MEMBUF_DEF_SIZE_INC** 20u

Functions

- void ixml_membuf_init (ixml_membuf *m)

  ixml_membuf initialization routine.
- void ixml_membuf_destroy (ixml_membuf *m)

  ixml_membuf clearing routine.
- int ixml_membuf_assign (ixml_membuf *m, const void *buf, size_t buf_len)

  Copies the contents o a buffer to the designated ixml_membuf.
- int ixml_membuf_assign_str (ixml_membuf *m, const char *c_str)

  *Copies a **NULL** terminated string to the ixml_buffer.*
- int ixml_membuf_append (ixml_membuf *m, const void *buf)

  Appends one byte to the designated ixml_membuffer.
- int ixml_membuf_append_str (ixml_membuf *m, const char *c_str)

  *Appends the contents of a **NULL** terminated string to the designated ixml_membuf.*
- int ixml_membuf_insert (ixml_membuf *m, const void *buf, size_t buf_len, size_t index)

9.8.1 Function Documentation

9.8.1.1 ixml_membuf_append()

```
int ixml_membuf_append (
            ixml_membuf * m,
            const void * buf )
```

Appends one byte to the designated ixml_membuffer.

This is a convenience function that internally uses ixml_membuf_insert().

Returns

The return value of ixml_membuf_insert().

See also

ixml_membuf_insert()

Parameters

in,out	*m*	The memory buffer on which to operate.
in	*buf*	The pointer to the byte to append.

References ixml_membuf_insert().

Referenced by copy_with_escape(), and Parser_appendTokBufChar().

9.8.1.2 ixml_membuf_append_str()

```
int ixml_membuf_append_str (
            ixml_membuf * m,
            const char * c_str )
```

Appends the contents of a **NULL** terminated string to the designated ixml_membuf.

This is a convenience function that internally uses ixml_membuf_insert().

Returns

The return value of ixml_membuf_insert().

See also

ixml_membuf_insert().

Parameters

in,out	*m*	The memory buffer on which to operate.
in	*c_str*	The input string to copy from.

References ixml_membuf_insert().

Referenced by copy_with_escape(), ixmlDocumenttoString(), ixmlDomTreetoString(), ixmlPrintDocument(), ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), and Parser_appendTokBufStr().

9.8.1.3 ixml_membuf_assign()

```
int ixml_membuf_assign (
            ixml_membuf * m,
            const void * buf,
            size_t buf_len )
```

Copies the contents o a buffer to the designated ixml_membuf.

The previous contents of the ixml_membuf are destroyed.

Returns

IXML_SUCCESS if successfull, or the error code returned by ixml_membuf_set_size().

See also

ixml_membuf_assign_str().

Parameters

in,out	*m*	The memory buffer on which to operate.
in	*buf*	The input buffer to copy from.
in	*buf_len*	The number of bytes to copy from the input buffer.

References ixml_membuf_destroy(), and ixml_membuf_set_size().

Referenced by ixml_membuf_assign_str().

9.8.1.4 ixml_membuf_assign_str()

```
int ixml_membuf_assign_str (
              ixml_membuf * m,
              const char * c_str )
```

Copies a **NULL** terminated string to the ixml_buffer.

This is a convenience function that internally uses ixml_membuf_assign().

Returns

The return value of ixml_membuf_assign().

See also

ixml_membuf_assign().

Parameters

in,out	*m*	The memory buffer on which to operate.
in	*c_str*	The input string to copy from.

References ixml_membuf_assign().

Referenced by Parser_setLastElem().

9.8.1.5 ixml_membuf_destroy()

```
void ixml_membuf_destroy (
              ixml_membuf * m )
```

ixml_membuf clearing routine.

The internal buffer is deleted and ixml_membuf_init() is called in the end to reinitialize the buffer.

Parameters

in,out	*m*	The memory buffer to clear.

References ixml_membuf_init().

Referenced by ixml_membuf_assign(), Parser_clearTokenBuf(), and Parser_free().

9.8.1.6 ixml_membuf_init()

```
void ixml_membuf_init (
             ixml_membuf * m )
```

ixml_membuf initialization routine.

Parameters

in,out	*m*	The memory buffer to initializa.

References MEMBUF_DEF_SIZE_INC.

Referenced by ixml_membuf_destroy(), ixmlDocumenttoString(), ixmlNodetoString(), ixmlPrintDocument(), ixmlPrintNode(), and Parser_init().

9.8.1.7 ixml_membuf_insert()

```
int ixml_membuf_insert (
             ixml_membuf * m,
             const void * buf,
             size_t buf_len,
             size_t index )
```

Returns

- 0 if successfull.
- IXML_INDEX_SIZE_ERR if the index parameter is out of range.
- Or the return code of ixml_membuf_set_size()

See also

ixml_membuf_set_size()

Parameters

in,out	*m*	The memory buffer on which to operate.
in	*buf*	The pointer to the input buffer.
in	*buf_len*	The buffer length.
in	*index*	The point of insertion relative to the beggining of the ixml_membuf internal buffer.

References ixml_membuf_set_size().

Referenced by ixml_membuf_append(), and ixml_membuf_append_str().

9.9 ixmlmembuf.h

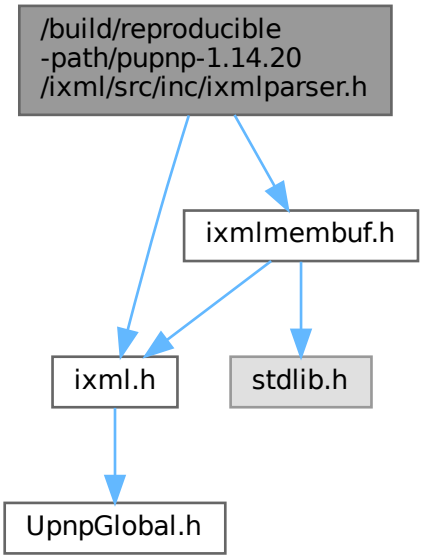
Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (c) 2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * - Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * - Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * - Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef IXML_MEMBUF_H
00034 #define IXML_MEMBUF_H
00035
00040 #include "ixml.h"
00041
00042 #include <stdlib.h> /* for size_t */
00043
00044 #define MINVAL(a, b) ((a) < (b) ? (a) : (b))
00045 #define MAXVAL(a, b) ((a) > (b) ? (a) : (b))
00046
00047 #define MEMBUF_DEF_SIZE_INC 20u
00048
00052 typedef struct
00053 {
00054         char *buf;
00055         size_t length;
00056         size_t capacity;
00057         size_t size_inc;
00058 } ixml_membuf;
00059
00063 void ixml_membuf_init(
00065         ixml_membuf *m);
00066
00073 void ixml_membuf_destroy(
00075         ixml_membuf *m);
00076
00087 int ixml_membuf_assign(
00089         ixml_membuf *m,
00091         const void *buf,
00093         size_t buf_len);
00094
00104 int ixml_membuf_assign_str(
00106         ixml_membuf *m,
00108         const char *c_str);
00109
00119 int ixml_membuf_append(
00121         ixml_membuf *m,
00123         const void *buf);
00124
00135 int ixml_membuf_append_str(
00137         ixml_membuf *m,
00139         const char *c_str);
00140
00151 int ixml_membuf_insert(
00153         ixml_membuf *m,
00155         const void *buf,
00157         size_t buf_len,
00160         size_t index);
00161
00162 #endif /* IXML_MEMBUF_H */
```

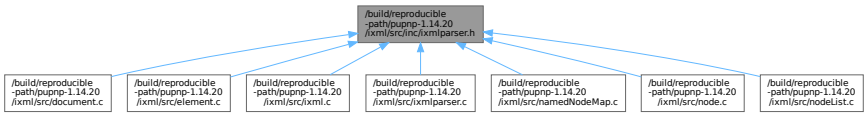
9.10 /build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlparser.h File Reference

```
#include "ixml.h"
#include "ixmlmembuf.h"
```

Include dependency graph for ixmlparser.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct _IXML_NamespaceURI
- struct _IXML_ElementStack
- struct _Parser

Macros

- #define **QUOT** """
- #define **LT** "<"
- #define **GT** ">"
- #define **APOS** "'"
- #define **AMP** "&"
- #define **ESC_HEX** "&#x"
- #define **ESC_DEC** "&#"

Typedefs

- typedef struct _IXML_NamespaceURI **IXML_NamespaceURI**
- typedef struct _IXML_ElementStack **IXML_ElementStack**
- typedef struct _Parser **Parser**

Enumerations

- enum **PARSER_STATE** { **eELEMENT** , **eATTRIBUTE** , **eCONTENT** }

Functions

- int Parser_isValidXmlName (const DOMString name)

  Check to see whether name is a valid xml name.
- void Parser_setErrorChar (char c)

  Sets the error character.
- void Parser_freeNodeContent (IXML_Node *IXML_Nodeptr)

  Fees a node contents.
- int Parser_LoadDocument (IXML_Document **retDoc, const char *xmlFile, int file)

  Parses a xml file and return the DOM tree.
- int Parser_setNodePrefixAndLocalName (IXML_Node *newIXML_NodeIXML_Attr)

  Set the node prefix and localName as defined by the nodeName in the form of ns:name.
- void **ixmlAttr_init** (IXML_Attr *attrNode)
- int ixmlElement_setTagName (IXML_Element *element, const char *tagName)

  Set the given element's tagName.
- void ixmlNamedNodeMap_init (IXML_NamedNodeMap *nnMap)

  Initializes a NamedNodeMap object.
- int ixmlNamedNodeMap_addToNamedNodeMap (IXML_NamedNodeMap **nnMap, IXML_Node *add)

  Add a node to a NamedNodeMap.
- int ixmlNodeList_addToNodeList (IXML_NodeList **nList, IXML_Node *add)

  Add a node to nodelist.
- void ixmlNode_init (IXML_Node *nodeptr)

  Intializes a node.
- int ixmlNode_compare (IXML_Node *srcNode, IXML_Node *destNode)

  Compare two nodes to see whether they are the same node. Parent, sibling and children node are ignored.
- void ixmlNode_getElementsByTagName (IXML_Node *n, const char *tagname, IXML_NodeList **list)

  Returns a nodeList of all descendant Elements with a given tagName, in the order in which they are encountered in a traversal of this element tree.
- void ixmlNode_getElementsByTagNameNS (IXML_Node *n, const char *namespaceURI, const char *localName, IXML_NodeList **list)

  Returns a nodeList of all the descendant Elements with a given local name and namespace URI in the order in which they are encountered in a preorder traversal of this Elememt tree.
- int ixmlNode_setNodeName (IXML_Node *node, const DOMString qualifiedName)
- int ixmlNode_setNodeProperties (IXML_Node *destNode, IXML_Node *src)
- void ixmlNodeList_init (IXML_NodeList *nList)

  Initializes a nodelist.

9.10.1 Function Documentation

9.10.1.1 ixmlElement_setTagName()

```
int ixmlElement_setTagName (
            IXML_Element * element,
            const char * tagName )
```

Set the given element's tagName.

Returns

One of the following:

- **IXML_SUCCESS**, if successfull.
- **IXML_FAILED**, if element of tagname is **NULL**.
- **IXML_INSUFFICIENT_MEMORY**, if there is no memory to allocate the buffer for the element's tagname.

Parameters

in	*element*	The element to change the tagname.
in	*tagName*	The new tagName for the element.

Referenced by ixmlNode_cloneElement().

9.10.1.2 ixmlNamedNodeMap_addToNamedNodeMap()

```
int ixmlNamedNodeMap_addToNamedNodeMap (
            IXML_NamedNodeMap ** nnMap,
            IXML_Node * add )
```

Add a node to a NamedNodeMap.

Returns

IXML_SUCCESS or failure.

References ixmlNamedNodeMap_init().

Referenced by ixmlNode_getAttributes().

9.10.1.3 ixmlNamedNodeMap_init()

```
void ixmlNamedNodeMap_init (
            IXML_NamedNodeMap * nnMap )
```

Initializes a NamedNodeMap object.

Parameters

in	*nnMap*	The named node map to process.

Referenced by ixmlNamedNodeMap_addToNamedNodeMap(), and ixmlNode_getAttributes().

9.10.1.4 ixmlNode_compare()

```
int ixmlNode_compare (
            IXML_Node * srcNode,
            IXML_Node * destNode )
```

Compare two nodes to see whether they are the same node. Parent, sibling and children node are ignored.

Returns

- 1, the two nodes are the same.
- 0, the two nodes are not the same.

Parameters

in	*srcNode*	The first **Node**.
in	*destNode*	The second **Node**.

Referenced by ixmlElement_findAttributeNode().

9.10.1.5 ixmlNode_getElementsByTagName()

```
void ixmlNode_getElementsByTagName (
            IXML_Node * n,
            const char * tagname,
            IXML_NodeList ** list )
```

Returns a nodeList of all descendant Elements with a given tagName, in the order in which they are encountered in a traversal of this element tree.

Parameters

in	*n*	The **Node** tree.
in	*tagname*	The tag name to match.
out	*list*	The output **NodeList**.

References ixmlNode_getElementsByTagNameRecursive(), ixmlNode_getFirstChild(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), and ixmlNodeList_addToNodeList().

9.10.1.6 ixmlNode_getElementsByTagNameNS()

```
void ixmlNode_getElementsByTagNameNS (
            IXML_Node * n,
```

```
            const char * namespaceURI,
            const char * localName,
            IXML_NodeList ** list )
```

Returns a nodeList of all the descendant Elements with a given local name and namespace URI in the order in which they are encountered in a preorder traversal of this Elememt tree.

Parameters

in	*n*	The **Element** tree.
in	*namespaceURI*	The name space to match.
in	*localName*	The local name to match.
out	*list*	The output **NodeList**.

References DOMString, ixmlNode_getElementsByTagNameNSRecursive(), ixmlNode_getFirstChild(), ixmlNode_getLocalName(), ixmlNode_getNamespaceURI(), ixmlNode_getNodeType(), and ixmlNodeList_addToNodeList().

9.10.1.7 ixmlNode_init()

```
void ixmlNode_init (
            IXML_Node * nodeptr )
```

Intializes a node.

Parameters

in	*nodeptr*	The **Node** to iniatialize.

Referenced by ixmlNode_cloneTextNode(), and Parser_parseDocument().

9.10.1.8 ixmlNode_setNodeName()

```
int ixmlNode_setNodeName (
            IXML_Node * node,
            const DOMString qualifiedName )
```

Returns

Parameters

in	*node*	The **Node**.
in	*qualifiedName*	.

References Parser_setNodePrefixAndLocalName().

Referenced by ixmlNode_cloneAttr(), ixmlNode_cloneCDATASect(), ixmlNode_cloneElement(), ixmlNode_cloneTextNode(), and ixmlNode_newDoc().

9.10.1.9 ixmlNode_setNodeProperties()

```
int ixmlNode_setNodeProperties (
            IXML_Node * destNode,
            IXML_Node * src )
```

Returns

Parameters

in	*destNode*	.
in	*src*	.

References ixmlNode_setLocalName(), ixmlNode_setNodeValue(), and ixmlNode_setPrefix().

Referenced by Parser_processAttributeName(), and Parser_processElementName().

9.10.1.10 ixmlNodeList_addToNodeList()

```
int ixmlNodeList_addToNodeList (
            IXML_NodeList ** nList,
            IXML_Node * add )
```

Add a node to nodelist.

Parameters

in	*nList*	The pointer to the nodelist.
in	*add*	The node to add.

References ixmlNodeList_init().

Referenced by ixmlNode_getChildNodes(), ixmlNode_getElementsByTagName(), ixmlNode_getElementsByTagNameNS(), ixmlNode_getElementsByTagNameNSRecursive(), and ixmlNode_getElementsByTagNameRecursive().

9.10.1.11 ixmlNodeList_init()

```
void ixmlNodeList_init (
            IXML_NodeList * nList )
```

Initializes a nodelist.

Parameters

in,out	*nList*	The **NodeList** to initialize.

Referenced by ixmlNode_getChildNodes(), and ixmlNodeList_addToNodeList().

9.10.1.12 Parser_freeNodeContent()

```
void Parser_freeNodeContent (
            IXML_Node * IXML_Nodeptr )
```

Fees a node contents.

Parameters

in	*IXML_Nodeptr*	The Node to process.

Referenced by Parser_parseDocument().

9.10.1.13 Parser_isValidXmlName()

```
int Parser_isValidXmlName (
            const DOMString name )
```

Check to see whether name is a valid xml name.

Parameters

in	*name*	The string to be checked.

References Parser_isNameChar().

9.10.1.14 Parser_LoadDocument()

```
int Parser_LoadDocument (
            IXML_Document ** retDoc,
            const char * xmlFile,
            int file )
```

Parses a xml file and return the DOM tree.

Parameters

out	*retDoc*	The output document tree.
in	*xmlFile*	The file name or the buffer to copy, according to the parameter "file".
in	*file*	1 if you want to read from a file, 0 if xmlFileName is the buffer to copy to the parser.

References _Parser::curPtr, _Parser::dataBuffer, Parser_free(), Parser_init(), Parser_parseDocument(), and Parser_readFileOrBuffer().

Referenced by ixmlLoadDocumentEx(), and ixmlParseBufferEx().

9.10.1.15 Parser_setErrorChar()

```
void Parser_setErrorChar (
            char c )
```

Sets the error character.

If 'c' is 0 (default), the parser is strict about XML encoding: invalid UTF-8 sequences or "&" entities are rejected, and the parsing aborts.

If 'c' is not 0, the parser is relaxed: invalid UTF-8 characters are replaced by this character, and invalid "&" entities are left untranslated. The parsing is then allowed to continue.

Parameters

in	*c*	The character to become the error character.

Referenced by ixmlRelaxParser().

9.10.1.16 Parser_setNodePrefixAndLocalName()

```
int Parser_setNodePrefixAndLocalName (
            IXML_Node * newIXML_NodeIXML_Attr )
```

Set the node prefix and localName as defined by the nodeName in the form of ns:name.

Parameters

in,out	*newIXML_NodeIXML_Attr*	The Node to process.

References safe_strdup().

Referenced by ixmlNode_setNodeName(), Parser_processSTag(), and Parser_xmlNamespace().

9.11 ixmlparser.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

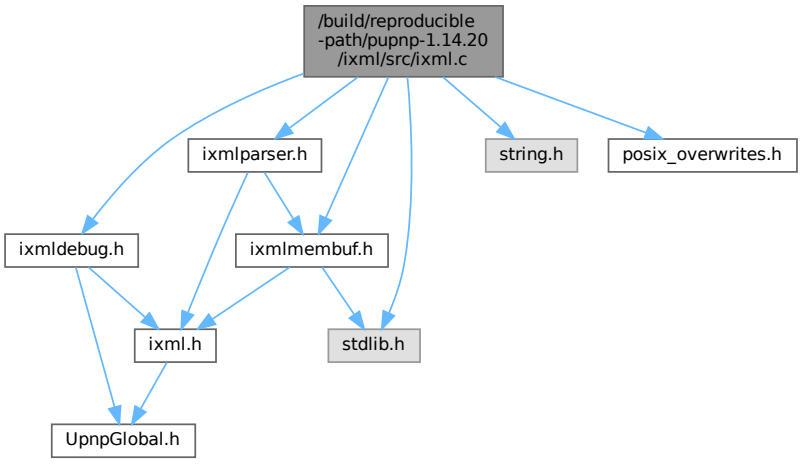
```
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef IXMLPARSER_H
00033 #define IXMLPARSER_H
00034
00039 #include "ixml.h"
00040 #include "ixmlmembuf.h"
00041
00042 /* Parser definitions */
00043 #define QUOT "&quot;"
00044 #define LT "&lt;"
00045 #define GT "&gt;"
00046 #define APOS "&apos;"
00047 #define AMP "&amp;"
00048 #define ESC_HEX "&#x"
00049 #define ESC_DEC "&#"
00050
00051 typedef struct _IXML_NamespaceURI
00052 {
00053         char *nsURI;
00054         char *prefix;
00055         struct _IXML_NamespaceURI *nextNsURI;
00056 } IXML_NamespaceURI;
00057
00058 typedef struct _IXML_ElementStack
00059 {
00060         char *element;
00061         char *prefix;
00062         char *namespaceUri;
00063         IXML_NamespaceURI *pNsURI;
00064         struct _IXML_ElementStack *nextElement;
00065 } IXML_ElementStack;
00066
00067 typedef enum
00068 {
00069         eELEMENT,
00070         eATTRIBUTE,
00071         eCONTENT
00072 } PARSER_STATE;
00073
00074 typedef struct _Parser
00075 {
00077         char *dataBuffer;
00079         char *curPtr;
00081         char *savePtr;
00082         ixml_membuf lastElem;
00083         ixml_membuf tokenBuf;
00084         IXML_Node *pNeedPrefixNode;
00085         IXML_ElementStack *pCurElement;
00086         IXML_Node *currentNodePtr;
00087         PARSER_STATE state;
00088         int bHasTopLevel;
00089 } Parser;
00090
00094 int Parser_isValidXmlName(
00096         const DOMString name);
00097
00109 void Parser_setErrorChar(
00111         char c);
00112
00113 #ifdef IXML_HAVE_SCRIPTSUPPORT
00122 void Parser_setBeforeFree(
00124         IXML_BeforeFreeNode_t hndlr);
00125
00129 IXML_BeforeFreeNode_t Parser_getBeforeFree(void);
00130 #endif
00131
00135 void Parser_freeNodeContent(
00137         IXML_Node *IXML_Nodeptr);
00138
00139 int Parser_LoadDocument(IXML_Document **retDoc, const char *xmlFile, int file);
00140
00141 int Parser_setNodePrefixAndLocalName(IXML_Node *newIXML_NodeIXML_Attr);
00142
00143 void ixmlAttr_init(IXML_Attr *attrNode);
00144
00154 int ixmlElement_setTagName(
00156         IXML_Element *element,
00158         const char *tagName);
00159
00163 void ixmlNamedNodeMap_init(
00165         IXML_NamedNodeMap *nnMap);
00166
00172 int ixmlNamedNodeMap_addToNamedNodeMap(
00173         /* [in] The named node map. */
00174         IXML_NamedNodeMap **nnMap,
```

```
00175         /* [in] The node to add. */
00176         IXML_Node *add);
00177
00181 int ixmlNodeList_addToNodeList(
00183         IXML_NodeList **nList,
00185         IXML_Node *add);
00186
00190 void ixmlNode_init(
00192         IXML_Node *nodeptr);
00193
00202 int ixmlNode_compare(
00204         IXML_Node *srcNode,
00206         IXML_Node *destNode);
00207
00213 void ixmlNode_getElementsByTagName(
00215         IXML_Node *n,
00217         const char *tagname,
00219         IXML_NodeList **list);
00220
00226 void ixmlNode_getElementsByTagNameNS(
00228         IXML_Node *n,
00230         const char *namespaceURI,
00232         const char *localName,
00234         IXML_NodeList **list);
00235
00241 int ixmlNode_setNodeName(
00243         IXML_Node *node,
00245         const DOMString qualifiedName);
00246
00252 int ixmlNode_setNodeProperties(
00254         IXML_Node *destNode,
00256         IXML_Node *src);
00257
00261 void ixmlNodeList_init(
00263         IXML_NodeList *nList);
00264
00265 #endif /* IXMLPARSER_H */
```

9.12 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixml.c File Reference

```
#include "ixmldebug.h"
#include "ixmlmembuf.h"
#include "ixmlparser.h"
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for ixml.c:



Functions

- static void copy_with_escape (ixml_membuf *buf, const char *p)

  Appends a string to a buffer, substituting some characters by escape sequences.
- static void ixmlPrintDomTreeRecursive (IXML_Node *nodeptr, ixml_membuf *buf)

  Recursive function to print all the node in a tree. Internal to parser only.
- static void ixmlPrintDomTree (IXML_Node *nodeptr, ixml_membuf *buf)

  Print a DOM tree.
- static void ixmlDomTreetoString (IXML_Node *nodeptr, ixml_membuf *buf)

  Converts a DOM tree into a text string.
- int ixmlLoadDocumentEx (const char *xmlFile, IXML_Document **doc)

  Parses an XML text file converting it into an IXML DOM representation.
- IXML_Document * ixmlLoadDocument (const char *xmlFile)

  Parses an XML text file converting it into an IXML DOM representation.
- DOMString ixmlPrintDocument (IXML_Document *doc)

  *Renders a **Node** and all sub-elements into an XML document representation.*
- DOMString ixmlPrintNode (IXML_Node *node)

  *Renders a **Node** and all sub-elements into an XML text representation.*
- DOMString ixmlDocumenttoString (IXML_Document *doc)

  *Renders a **Node** and all sub-elements into an XML document representation.*
- DOMString ixmlNodetoString (IXML_Node *node)

  *Renders a **Node** and all sub-elements into an XML text representation. The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.*
- void ixmlRelaxParser (char errorChar)

  Makes the XML parser more tolerant to malformed text.
- int ixmlParseBufferEx (const char *buffer, IXML_Document **retDoc)

  Parses an XML text buffer converting it into an IXML DOM representation.
- IXML_Document * ixmlParseBuffer (const char *buffer)

  Parses an XML text buffer converting it into an IXML DOM representation.
- DOMString **ixmlCloneDOMString** (const DOMString src)
- void **ixmlFreeDOMString** (DOMString buf)

9.12.1 Function Documentation

9.12.1.1 copy_with_escape()

```
static void copy_with_escape (
            ixml_membuf * buf,
            const char * p )  [static]
```

Appends a string to a buffer, substituting some characters by escape sequences.

Parameters

in,out	*buf*	The input/output buffer.
in	*p*	The string to copy from.

References ixml_membuf_append(), and ixml_membuf_append_str().

Referenced by ixmlDomTreetoString(), ixmlPrintDomTree(), and ixmlPrintDomTreeRecursive().

9.12.1.2 ixmlDomTreetoString()

```
static void ixmlDomTreetoString (
             IXML_Node * nodeptr,
             ixml_membuf * buf )  [static]
```

Converts a DOM tree into a text string.

Element, and Attribute nodes are handled differently. We don't want to print the Element and Attribute nodes' sibling.

Parameters

in	*nodeptr*	

Todo documentation.

Parameters

in	*buf*	

Todo documentation.

References copy_with_escape(), ixml_membuf_append_str(), ixmlNode_getFirstChild(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), ixmlNode_getNodeValue(), ixmlPrintDomTreeRecursive(), and IxmlPrintf().

Referenced by ixmlDocumenttoString(), and ixmlNodetoString().

9.12.1.3 ixmlPrintDomTree()

```
static void ixmlPrintDomTree (
             IXML_Node * nodeptr,
             ixml_membuf * buf )  [static]
```

Print a DOM tree.

Element, and Attribute nodes are handled differently. We don't want to print the Element and Attribute nodes' sibling.

Parameters

in	*nodeptr*	

Todo documentation.

Parameters

in	*buf*	

Todo documentation.

References copy_with_escape(), ixml_membuf_append_str(), ixmlNode_getFirstChild(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), ixmlNode_getNodeValue(), ixmlPrintDomTreeRecursive(), and IxmlPrintf().

Referenced by ixmlPrintDocument(), and ixmlPrintNode().

9.12.1.4 ixmlPrintDomTreeRecursive()

```
static void ixmlPrintDomTreeRecursive (
              IXML_Node * nodeptr,
              ixml_membuf * buf )  [static]
```

Recursive function to print all the node in a tree. Internal to parser only.

Parameters

in	*nodeptr*	

Todo documentation.

Parameters

in	*buf*	

Todo documentation.

References copy_with_escape(), ixml_membuf_append_str(), ixmlNode_getFirstChild(), ixmlNode_getNextSibling(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), ixmlNode_getNodeValue(), ixmlPrintDomTreeRecursive(), and IxmlPrintf().

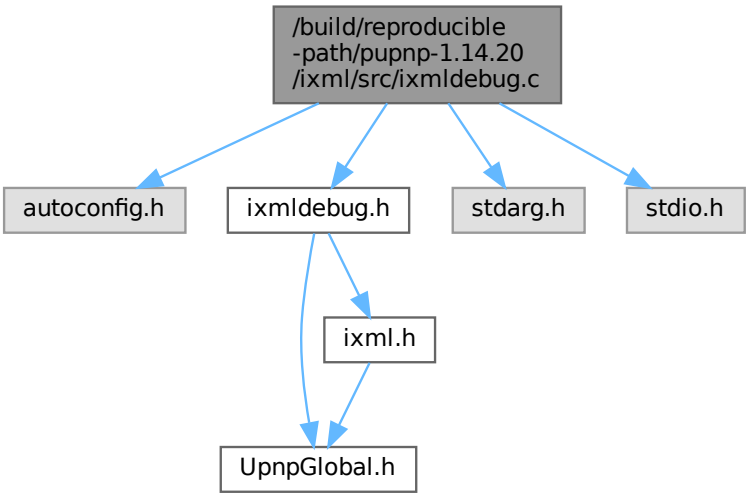
Referenced by ixmlDomTreetoString(), ixmlPrintDomTree(), and ixmlPrintDomTreeRecursive().

9.13 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixmldebug.c File Reference

```
#include "autoconfig.h"
#include "ixmldebug.h"
#include <stdarg.h>
```

```
#include <stdio.h>
```

Include dependency graph for ixmldebug.c:



Functions

- void IxmlPrintf (const char *DbgFileName, int DbgLineNo, const char *FunctionName, const char *FmtStr,...)

  Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.
- void printNodes (IXML_Node *tmpRoot, int depth)

  Print the node names and values of a XML tree.

9.13.1 Function Documentation

9.13.1.1 IxmlPrintf()

```
void IxmlPrintf (
            const char * DbgFileName,
            int DbgLineNo,
            const char * FunctionName,
            const char * FmtStr,
             ... )
```

Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.

Parameters

in	*DbgFileName*	The file name, usually **FILE**.
in	*DbgLineNo*	The line number, usually **LINE** or a variable that got the **LINE** at the appropriate place.
in	*FunctionName*	The function name.
in	*FmtStr*	Printf like format specification.
in	...	Printf like Variable number of arguments that will go in the debug statement.

Referenced by ixmlDomTreetoString(), ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), Parser_copyToken(), Parser_getChar(), Parser_getNextNode(), Parser_processAttribute(), Parser_processContent(), Parser_processETag(), Parser_UTF8ToInt(), Parser_xmlNamespace(), and printNodes().

9.13.1.2 printNodes()

```
void printNodes (
              IXML_Node * tmpRoot,
              int depth )
```

Print the node names and values of a XML tree.

Parameters

in	*tmpRoot*	The root of the tree to print.
in	*depth*	The depth to print.

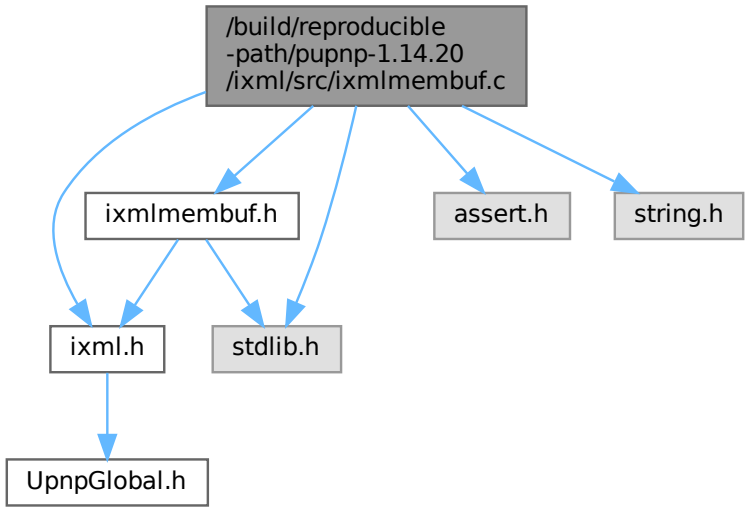
References DOMString, ixmlNode_getChildNodes(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), ixmlNode_getNodeValue(), ixmlNodeList_item(), IxmlPrintf(), and printNodes().

Referenced by printNodes().

9.14 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlmembuf.c File Reference

```
#include "ixmlmembuf.h"
#include "ixml.h"
#include <assert.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for ixmlmembuf.c:



Functions

- static int ixml_membuf_set_size (ixml_membuf *m, size_t new_length)

  Increases or decreases buffer capacity so that at least 'new_length' bytes can be stored.
- void ixml_membuf_init (ixml_membuf *m)

  ixml_membuf initialization routine.
- void ixml_membuf_destroy (ixml_membuf *m)

  ixml_membuf clearing routine.
- int ixml_membuf_assign (ixml_membuf *m, const void *buf, size_t buf_len)

  Copies the contents o a buffer to the designated ixml_membuf.
- int ixml_membuf_assign_str (ixml_membuf *m, const char *c_str)

  *Copies a **NULL** terminated string to the ixml_buffer.*
- int ixml_membuf_append (ixml_membuf *m, const void *buf)

  Appends one byte to the designated ixml_membuffer.
- int ixml_membuf_append_str (ixml_membuf *m, const char *c_str)

  *Appends the contents of a **NULL** terminated string to the designated ixml_membuf.*
- int ixml_membuf_insert (ixml_membuf *m, const void *buf, size_t buf_len, size_t index)

9.14.1 Function Documentation

9.14.1.1 ixml_membuf_append()

```
int ixml_membuf_append (
            ixml_membuf * m,
            const void * buf )
```

Appends one byte to the designated ixml_membuffer.

This is a convenience function that internally uses ixml_membuf_insert().

Returns

The return value of ixml_membuf_insert().

See also

ixml_membuf_insert()

Parameters

in,out	*m*	The memory buffer
in	*buf*	The buffer to append

References ixml_membuf_insert().

Referenced by copy_with_escape(), and Parser_appendTokBufChar().

9.14.1.2 ixml_membuf_append_str()

```
int ixml_membuf_append_str (
```

```
            ixml_membuf * m,
            const char * c_str )
```

Appends the contents of a **NULL** terminated string to the designated ixml_membuf.

This is a convenience function that internally uses ixml_membuf_insert().

Returns

The return value of ixml_membuf_insert().

See also

ixml_membuf_insert().

Parameters

in,out	*m*	The memory buffer
in	*c_str*	The characters to append (null-terminated)

References ixml_membuf_insert().

Referenced by copy_with_escape(), ixmlDocumenttoString(), ixmlDomTreetoString(), ixmlPrintDocument(), ixmlPrintDomTree(), ixmlPrintDomTreeRecursive(), and Parser_appendTokBufStr().

9.14.1.3 ixml_membuf_assign()

```
int ixml_membuf_assign (
            ixml_membuf * m,
            const void * buf,
            size_t buf_len )
```

Copies the contents o a buffer to the designated ixml_membuf.

The previous contents of the ixml_membuf are destroyed.

Returns

IXML_SUCCESS if successfull, or the error code returned by ixml_membuf_set_size().

See also

ixml_membuf_assign_str().

Parameters

in,out	*m*	The memory buffer on which to operate.
in	*buf*	The input buffer to copy from.
in	*buf_len*	The number of bytes to copy from the input buffer.

References ixml_membuf_destroy(), and ixml_membuf_set_size().

Referenced by ixml_membuf_assign_str().

9.14.1.4 ixml_membuf_assign_str()

```
int ixml_membuf_assign_str (
            ixml_membuf * m,
            const char * c_str )
```

Copies a **NULL** terminated string to the ixml_buffer.

This is a convenience function that internally uses ixml_membuf_assign().

Returns

The return value of ixml_membuf_assign().

See also

ixml_membuf_assign().

Parameters

in,out	*m*	The memory buffer on which to operate.
in	*c_str*	The input string to copy from.

References ixml_membuf_assign().

Referenced by Parser_setLastElem().

9.14.1.5 ixml_membuf_destroy()

```
void ixml_membuf_destroy (
            ixml_membuf * m )
```

ixml_membuf clearing routine.

The internal buffer is deleted and ixml_membuf_init() is called in the end to reinitialize the buffer.

Parameters

in,out	*m*	The memory buffer to clear.

References ixml_membuf_init().

Referenced by ixml_membuf_assign(), Parser_clearTokenBuf(), and Parser_free().

9.14.1.6 ixml_membuf_init()

```
void ixml_membuf_init (
             ixml_membuf * m )
```

ixml_membuf initialization routine.

Parameters

in,out	*m*	The memory buffer to initializa.

References MEMBUF_DEF_SIZE_INC.

Referenced by ixml_membuf_destroy(), ixmlDocumenttoString(), ixmlNodetoString(), ixmlPrintDocument(), ixmlPrintNode(), and Parser_init().

9.14.1.7 ixml_membuf_insert()

```
int ixml_membuf_insert (
             ixml_membuf * m,
             const void * buf,
             size_t buf_len,
             size_t index )
```

Returns

- 0 if successfull.
- IXML_INDEX_SIZE_ERR if the index parameter is out of range.
- Or the return code of ixml_membuf_set_size()

See also

ixml_membuf_set_size()

Parameters

in,out	*m*	The memory buffer
in	*buf*	The buffer to insert
in	*buf_len*	The length of the buffer
in	*index*	Position where to insert the buffer

References ixml_membuf_set_size().

Referenced by ixml_membuf_append(), and ixml_membuf_append_str().

9.14.1.8 ixml_membuf_set_size()

```
static int ixml_membuf_set_size (
             ixml_membuf * m,
             size_t new_length )  [static]
```

Increases or decreases buffer capacity so that at least 'new_length' bytes can be stored.

On error, m's fields do not change.

Returns

- UPNP_E_SUCCESS
- UPNP_E_OUTOF_MEMORY

Parameters

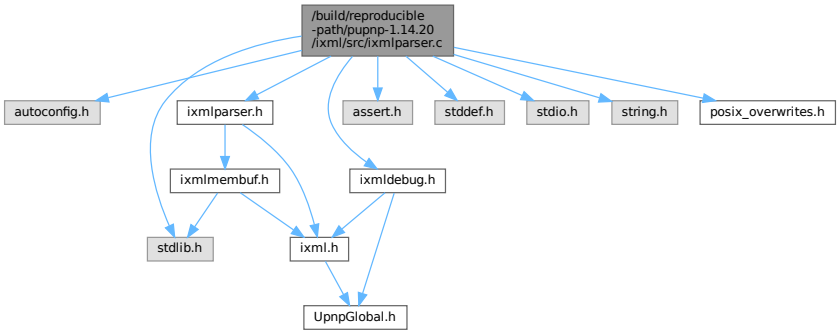
in,out	*m*	The memory buffer.
in	*new_length*	The new lenght.

Referenced by ixml_membuf_assign(), and ixml_membuf_insert().

9.15 /build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlparser.c File Reference

```
#include "autoconfig.h"
#include "ixmlparser.h"
#include "ixmldebug.h"
#include <assert.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for ixmlparser.c:



Data Structures

- struct char_info

Macros

- #define LETTERTABLESIZE (sizeof(Letter) / sizeof(Letter[0]))

  The size of the letter table array.
- #define NAMECHARTABLESIZE (sizeof(NameChar) / sizeof(NameChar[0]))

  The name char table array size.

Typedefs

- typedef struct char_info **char_info_t**
- typedef char **utf8char**[8]

Functions

- static void Parser_freeElementStackItem (IXML_ElementStack *pItem)

  Frees one ElementStack item.
- static void Parser_freeNsURI (IXML_NamespaceURI *pNsURI)

  Frees namespaceURI item.
- static void Parser_free (Parser *xmlParser)

  Frees all temporary memory allocated by xmlparser.
- static int Parser_skipDocType (char **pstr)

  Skips document type declaration.
- static int Parser_skipString (char **pstrSrc, const char *strSkipKey)

  Skips all characters in the string until it finds the skip key. Then it skips the skip key and returns.
- static void Parser_skipBom (Parser *xmlParser)

  Skip UTF-8 byte order mark.
- static void Parser_skipWhiteSpaces (Parser *xmlParser)

  Skip white spaces.
- static int Parser_skipXMLDecl (Parser *xmlParser)

  Skips XML declarations.
- static int Parser_skipComment (char **pstrSrc)

  Skips all characters in the string until it finds the skip key. Then it skips the skip key and returns.
- static int Parser_skipMisc (Parser *xmlParser)

  Skip comment, PI and white space.
- static int Parser_skipProlog (Parser *xmlParser)

  Skip prolog.
- static int Parser_setLastElem (Parser *xmlParser, const char *s)

  Set the last element to be the given string.
- static void Parser_clearTokenBuf (Parser *xmlParser)

  Clear token buffer.
- static int Parser_UTF8ToInt (const char *ss, ptrdiff_t *len)

  In UTF-8, characters are encoded using sequences of 1 to 6 octets. This functions will return a UTF-8 character value and its octets number.
- static int Parser_isCharInTable (int c, char_info_t *tbl, int sz)

  Will determine whether character c is in the table of tbl (either Letter table or NameChar table).
- static int Parser_isNameChar (int c, int bNameChar)

  Check whether c (int) is in LetterTable or NameCharTable.
- static int Parser_isXmlChar (int c)

  see XML 1.0 (2nd Edition) 2.2.
- static int Parser_getChar (const char *src, ptrdiff_t *cLen)

Returns next char value and its length.

- static int Parser_appendTokBufChar (Parser *xmlParser, char c)

  Appends c to token buffer.
- static int Parser_intToUTF8 (int c, utf8char s)

  Encodes a character to its UTF-8 character string, and return its length.
- static int Parser_appendTokBufStr (Parser *xmlParser, const char *s)

  Appends string s to token buffer.
- static int Parser_copyToken (Parser *xmlParser, const char *src, ptrdiff_t len)

  Copy string in src into xml parser token buffer.
- static ptrdiff_t Parser_getNextToken (Parser *xmlParser)

  Return the length of next token in tokenBuff.
- static char * safe_strdup (const char *s)

  Version of strdup() that handles NULL input.
- static int Parser_processSTag (Parser *xmlParser, IXML_Node *node)

  Processes the STag as defined by XML spec.
- static int Parser_skipPI (char **pSrc)
- static int Parser_processCDSect (char **pSrc, IXML_Node *node)

  Processes CDSection as defined by XML spec.
- static int Parser_processContent (Parser *xmlParser, IXML_Node *node)

  Processes the CONTENT as defined in XML spec.
- static int Parser_processETag (Parser *xmlParser, IXML_Node *node, int *bETag)

  Process ETag as defined by XML spec.
- static char * Parser_getNameSpace (Parser *xmlParser, const char *prefix)

  Unimplemented function.
- static int Parser_addNamespace (Parser *xmlParser)

  Add a namespace definition.
- static int Parser_xmlNamespace (Parser *xmlParser, IXML_Node *newNode)

  Add namespace definition.
- static int Parser_processAttribute (Parser *xmlParser, IXML_Node *node)

  Processes attribute.
- static int Parser_getNextNode (Parser *xmlParser, IXML_Node *node, int *bETag)

  Get the next node.
- static int Parser_ElementPrefixDefined (Parser *xmlParser, IXML_Node *newNode, char **nsURI)

  Decides whether element's prefix is already defined.
- static int Parser_setElementNamespace (IXML_Element *newElement, const char *nsURI)

  Set element's namespace.
- static int isDuplicateAttribute (Parser *xmlParser, IXML_Node *newAttrNode)

  Reports whether the new attribute is the same as an existing one.
- static int Parser_processAttributeName (IXML_Document *rootDoc, Parser *xmlParser, IXML_Node *new↩ Node)

  Processes the attribute name.
- static int Parser_pushElement (Parser *xmlParser, IXML_Node *newElement)

  Push a new element onto element stack.
- static int isTopLevelElement (Parser *xmlParser)

  Reports whether there is a top level element in the parser.
- static int Parser_hasDefaultNamespace (Parser *xmlParser, char **nsURI)

  Decide whether the current element has default namespace.
- static int Parser_processElementName (IXML_Document *rootDoc, Parser *xmlParser, IXML_Node *new↩ Node)

  Processes element name.
- static int Parser_isValidEndElement (Parser *xmlParser, IXML_Node *newNode)

Check if a new node->nodeName matches top of element stack.

- static void Parser_popElement (Parser *xmlParser)

  Remove element from element stack.

- static int Parser_eTagVerification (Parser *xmlParser, IXML_Node *newNode)

  Verifies endof element tag is the same as the openning element tag.

- static int Parser_parseDocument (IXML_Document **retDoc, Parser *xmlParser)

  Parses the xml file and returns the DOM document tree.

- int Parser_isValidXmlName (const DOMString name)

  Check to see whether name is a valid xml name.

- void Parser_setErrorChar (char c)

  Sets the error character.

- static Parser * Parser_init (void)

  Initializes a xml parser.

- static int Parser_readFileOrBuffer (Parser *xmlParser, const char *xmlFileName, int file)

  Read a xml file or buffer contents into xml parser.

- int Parser_LoadDocument (IXML_Document **retDoc, const char *xmlFileName, int file)

  Parses a xml file and return the DOM tree.

- void Parser_freeNodeContent (IXML_Node *nodeptr)

  Fees a node contents.

- int Parser_setNodePrefixAndLocalName (IXML_Node *node)

  Set the node prefix and localName as defined by the nodeName in the form of ns:name.

Variables

- static char **g_error_char** = '\0'
- static const char **LESSTHAN** = '<'
- static const char **GREATERTHAN** = '>'
- static const char **SLASH** = '/'
- static const char **EQUALS** = '='
- static const char **QUOTE** = '\"'
- static const char **SINGLEQUOTE** = '\''
- static const char * **WHITESPACE** = "\n\t\r "
- static const char * **COMPLETETAG** = "/>"
- static const char * **ENDTAG** = "</"
- static const char * **XMLDECL** = "<?xml "
- static const char * **XMLDECL2** = "<?xml?"
- static const char * **BEGIN_COMMENT** = "<!--"
- static const char * **END_COMMENT** = "-->"
- static const char * **BEGIN_PI** = "<?"
- static const char * **END_PI** = "?>"
- static const char * **BEGIN_DOCTYPE** = "<!DOCTYPE"
- static const char * **CDSTART** = "<![CDATA["
- static const char * **CDEND** = "]]>"
- static const char * **DEC_NUMBERS** = "0123456789"
- static const char * **HEX_NUMBERS** = "0123456789ABCDEFabcdef"
- static const char * **UTF8_BOM** = "\xef\xbb\xbf"
- static char_info_t Letter [ ]

  The letter table contains all characters in XML 1.0 plus ":", "_" and ideographic.

- static char_info_t NameChar [ ]

  The NameChar table contains CombiningChar, Extender, Digit, '-', '.', less '_', ':'.

9.15.1 Macro Definition Documentation

9.15.1.1 LETTERTABLESIZE

```
#define LETTERTABLESIZE (sizeof(Letter) / sizeof(Letter[0]))
```

The size of the letter table array.

9.15.1.2 NAMECHARTABLESIZE

```
#define NAMECHARTABLESIZE (sizeof(NameChar) / sizeof(NameChar[0]))
```

The name char table array size.

9.15.2 Function Documentation

9.15.2.1 isDuplicateAttribute()

```
static int isDuplicateAttribute (
            Parser * xmlParser,
            IXML_Node * newAttrNode )  [static]
```

Reports whether the new attribute is the same as an existing one.

Returns

1 if the new attribute is the same as an existing one.

Parameters

in	*xmlParser*	The XML parser.
in	*newAttrNode*	The node attribute to compare.

Referenced by Parser_processAttributeName().

9.15.2.2 isTopLevelElement()

```
static int isTopLevelElement (
            Parser * xmlParser )  [static]
```

Reports whether there is a top level element in the parser.

Returns

1 if there is a top level element in the parser.

Parameters

in	*xmlParser*	The XML parser.

Referenced by Parser_processElementName().

9.15.2.3 Parser_addNamespace()

```
static int Parser_addNamespace (
            Parser * xmlParser ) [static]
```

Add a namespace definition.

Parameters

in	*xmlParser*	The XML parser.

References Parser_getNameSpace(), and safe_strdup().

Referenced by Parser_xmlNamespace().

9.15.2.4 Parser_appendTokBufChar()

```
static int Parser_appendTokBufChar (
            Parser * xmlParser,
            char c ) [static]
```

Appends c to token buffer.

Parameters

in	*xmlParser*	The XML parser.
in	*c*	The character to append.

References ixml_membuf_append().

Referenced by Parser_copyToken().

9.15.2.5 Parser_appendTokBufStr()

```
static int Parser_appendTokBufStr (
            Parser * xmlParser,
            const char * s ) [static]
```

Appends string s to token buffer.

Parameters

in	*xmlParser*	The XML parser.
in	*s*	The string to append.

References ixml_membuf_append_str().

Referenced by Parser_copyToken().

9.15.2.6 Parser_clearTokenBuf()

```
static void Parser_clearTokenBuf (
            Parser * xmlParser )  [static]
```

Clear token buffer.

Parameters

in	*xmlParser*	The XML parser.

References ixml_membuf_destroy().

Referenced by Parser_getNextToken(), Parser_processAttribute(), and Parser_processContent().

9.15.2.7 Parser_copyToken()

```
static int Parser_copyToken (
            Parser * xmlParser,
            const char * src,
            ptrdiff_t len )  [static]
```

Copy string in src into xml parser token buffer.

Parameters

in	*xmlParser*	The XML parser.
in	*src*	The string to copy from.
in	*len*	The lenght to copy.

References IxmlPrintf(), Parser_appendTokBufChar(), Parser_appendTokBufStr(), Parser_getChar(), and Parser_intToUTF8().

Referenced by Parser_getNextToken(), Parser_processAttribute(), and Parser_processContent().

9.15.2.8 Parser_ElementPrefixDefined()

```
static int Parser_ElementPrefixDefined (
            Parser * xmlParser,
```

```
            IXML_Node * newNode,
            char ** nsURI )  [static]
```

Decides whether element's prefix is already defined.

Parameters

in	*xmlParser*	The XML parser.
in	*newNode*	The Node to process.
in,out	*nsURI*	The name space URI.

Referenced by Parser_processElementName().

9.15.2.9 Parser_eTagVerification()

```
static int Parser_eTagVerification (
            Parser * xmlParser,
            IXML_Node * newNode )  [static]
```

Verifies endof element tag is the same as the openning element tag.

Parameters

in	*xmlParser*	The XML parser.
in	*newNode*	The Node to process.

References Parser_isValidEndElement(), and Parser_popElement().

Referenced by Parser_parseDocument().

9.15.2.10 Parser_free()

```
static void Parser_free (
            Parser * xmlParser )  [static]
```

Frees all temporary memory allocated by xmlparser.

Parameters

in	*xmlParser*	The XML parser.

References _Parser::dataBuffer, ixml_membuf_destroy(), Parser_freeElementStackItem(), and Parser_freeNsURI().

Referenced by Parser_LoadDocument(), and Parser_parseDocument().

9.15.2.11 Parser_freeElementStackItem()

```
static void Parser_freeElementStackItem (
            IXML_ElementStack * pItem )  [static]
```

Frees one ElementStack item.

Parameters

in	*pItem*	The element stack item to free.

Referenced by Parser_free(), Parser_popElement(), and Parser_pushElement().

9.15.2.12 Parser_freeNodeContent()

```
void Parser_freeNodeContent (
            IXML_Node * nodeptr )
```

Fees a node contents.

Parameters

in	*nodeptr*	The Node to process.

Referenced by Parser_parseDocument().

9.15.2.13 Parser_freeNsURI()

```
static void Parser_freeNsURI (
            IXML_NamespaceURI * pNsURI )  [static]
```

Frees namespaceURI item.

Parameters

in	*pNsURI*	The name space URI item to free.

Referenced by Parser_free(), Parser_popElement(), and Parser_xmlNamespace().

9.15.2.14 Parser_getChar()

```
static int Parser_getChar (
            const char * src,
            ptrdiff_t * cLen )  [static]
```

Returns next char value and its length.

Parameters

in	*src*	.
in,out	*cLen*	.

References IxmlPrintf(), Parser_isXmlChar(), and Parser_UTF8ToInt().

Referenced by Parser_copyToken().

9.15.2.15 Parser_getNameSpace()

```
static char * Parser_getNameSpace (
            Parser * xmlParser,
            const char * prefix )  [static]
```

Unimplemented function.

Returns

IXML_SUCCESS.

Return the namespce as defined as prefix.

Parameters

in	*xmlParser*	The XML parser.
in	*prefix*	The prefix.

Referenced by Parser_addNamespace().

9.15.2.16 Parser_getNextNode()

```
static int Parser_getNextNode (
            Parser * xmlParser,
            IXML_Node * node,
            int * bETag )  [static]
```

Get the next node.

Returns

IXML_SUCCESS and the next node or IXML_FILE_DONE or an error.

Parameters

in	*xmlParser*	The XML parser.
out	*node*	The XML parser.
out	*bETag*	The XML parser.

References _Parser::curPtr, IxmlPrintf(), Parser_getNextToken(), Parser_processAttribute(), Parser_processContent(), Parser_processETag(), Parser_processSTag(), Parser_skipWhiteSpaces(), and safe_strdup().

Referenced by Parser_parseDocument().

9.15.2.17 Parser_getNextToken()

```
static ptrdiff_t Parser_getNextToken (
            Parser * xmlParser )  [static]
```

Return the length of next token in tokenBuff.

Parameters

in	*xmlParser*	The XML parser.

References _Parser::curPtr, Parser_clearTokenBuf(), Parser_copyToken(), Parser_isNameChar(), Parser_skipMisc(), Parser_UTF8ToInt(), and _Parser::savePtr.

Referenced by Parser_getNextNode(), Parser_processAttribute(), Parser_processETag(), and Parser_processSTag().

9.15.2.18 Parser_hasDefaultNamespace()

```
static int Parser_hasDefaultNamespace (
            Parser * xmlParser,
            char ** nsURI )  [static]
```

Decide whether the current element has default namespace.

Parameters

in	*xmlParser*	The XML parser.
in,out	*nsURI*	The name space URI.

Referenced by Parser_processElementName().

9.15.2.19 Parser_init()

```
static Parser * Parser_init (
            void  )  [static]
```

Initializes a xml parser.

Returns

The parser object or **NULL** if there is not enough memory.

References ixml_membuf_init().

Referenced by Parser_LoadDocument().

9.15.2.20 Parser_intToUTF8()

```
static int Parser_intToUTF8 (
            int c,
            utf8char s )  [static]
```

Encodes a character to its UTF-8 character string, and return its length.

Returns

The length of the encoded string in bytes.

Parameters

in	*c*	The character to encode.
out	*s*	The resultant UTF-8 encoded string.

Referenced by Parser_copyToken().

9.15.2.21 Parser_isCharInTable()

```
static int Parser_isCharInTable (
            int c,
            char_info_t * tbl,
            int sz )  [static]
```

Will determine whether character c is in the table of tbl (either Letter table or NameChar table).

Returns

1 or 0.

Parameters

in	*c*	Character to check.
in	*tbl*	Table to use.
in	*sz*	Size of the table.

Referenced by Parser_isNameChar().

9.15.2.22 Parser_isNameChar()

```
static int Parser_isNameChar (
            int c,
            int bNameChar )  [static]
```

Check whether c (int) is in LetterTable or NameCharTable.

Parameters

in	*c*	The character to check.
in	*bNameChar*	1 if you also want to check in the NameChar table.

References Letter, LETTERTABLESIZE, NameChar, NAMECHARTABLESIZE, and Parser_isCharInTable().

Referenced by Parser_getNextToken(), Parser_isValidXmlName(), and Parser_processAttribute().

9.15.2.23 Parser_isValidEndElement()

```
static int Parser_isValidEndElement (
            Parser * xmlParser,
            IXML_Node * newNode )  [static]
```

Check if a new node->nodeName matches top of element stack.

Returns

1 if matches.

Parameters

in	*xmlParser*	The XML parser.
in	*newNode*	The node.

Referenced by Parser_eTagVerification().

9.15.2.24 Parser_isValidXmlName()

```
int Parser_isValidXmlName (
            const DOMString name )
```

Check to see whether name is a valid xml name.

Parameters

in	*name*	The string to be checked.

References Parser_isNameChar().

9.15.2.25 Parser_isXmlChar()

```
static int Parser_isXmlChar (
            int c )  [static]
```

see XML 1.0 (2nd Edition) 2.2.

Parameters

in	*c*	The character to check.

Referenced by Parser_getChar(), and Parser_processCDSect().

9.15.2.26 Parser_LoadDocument()

```
int Parser_LoadDocument (
            IXML_Document ** retDoc,
            const char * xmlFileName,
            int file )
```

Parses a xml file and return the DOM tree.

Parameters

out	*retDoc*	The output document tree.
in	*xmlFileName*	The file name or the buffer to copy, according to the parameter "file".
in	*file*	1 if you want to read from a file, 0 if xmlFileName is the buffer to copy to the parser.

References _Parser::curPtr, _Parser::dataBuffer, Parser_free(), Parser_init(), Parser_parseDocument(), and Parser_readFileOrBuffer().

Referenced by ixmlLoadDocumentEx(), and ixmlParseBufferEx().

9.15.2.27 Parser_parseDocument()

```
static int Parser_parseDocument (
            IXML_Document ** retDoc,
            Parser * xmlParser )  [static]
```

Parses the xml file and returns the DOM document tree.

Returns

Parameters

out	*retDoc*	The XML document.
in	*xmlParser*	The XML parser.

References ixmlDocument_createCDATASectionEx(), ixmlDocument_createDocumentEx(), ixmlDocument_createTextNodeEx(), ixmlDocument_free(), ixmlNode_appendChild(), ixmlNode_init(), Parser_eTagVerification(), Parser_free(), Parser_freeNodeContent(), Parser_getNextNode(), Parser_processAttributeName(), Parser_processElementName(), and Parser_skipProlog().

Referenced by Parser_LoadDocument().

9.15.2.28 Parser_popElement()

```
static void Parser_popElement (
            Parser * xmlParser )  [static]
```

Remove element from element stack.

Parameters

in	*xmlParser*	The XML parser.

References Parser_freeElementStackItem(), and Parser_freeNsURI().

Referenced by Parser_eTagVerification().

9.15.2.29 Parser_processAttribute()

```
static int Parser_processAttribute (
            Parser * xmlParser,
            IXML_Node * node )  [static]
```

Processes attribute.

Returns

IXML_SUCCESS or failure code.

Parameters

in	*xmlParser*	The XML parser.
in	*node*	The Node to process.

References _Parser::curPtr, IxmlPrintf(), Parser_clearTokenBuf(), Parser_copyToken(), Parser_getNextToken(), Parser_isNameChar(), Parser_UTF8ToInt(), Parser_xmlNamespace(), safe_strdup(), and _Parser::savePtr.

Referenced by Parser_getNextNode().

9.15.2.30 Parser_processAttributeName()

```
static int Parser_processAttributeName (
            IXML_Document * rootDoc,
            Parser * xmlParser,
            IXML_Node * newNode )  [static]
```

Processes the attribute name.

Returns

IXML_SUCCESS if successful, otherwise or an error code.

Parameters

in	*rootDoc*	The XML document.
in	*xmlParser*	The XML parser.
in	*newNode*	The Node to process.

References isDuplicateAttribute(), ixmlAttr_free(), ixmlDocument_createAttributeEx(), ixmlElement_setAttributeNode(), and ixmlNode_setNodeProperties().

Referenced by Parser_parseDocument().

9.15.2.31 Parser_processCDSect()

```
static int Parser_processCDSect (
            char ** pSrc,
            IXML_Node * node )  [static]
```

Processes CDSection as defined by XML spec.

Returns

Parameters

in	*pSrc*	.
in	*node*	The Node to process.

References Parser_isXmlChar(), and safe_strdup().

Referenced by Parser_processContent().

9.15.2.32 Parser_processContent()

```
static int Parser_processContent (
            Parser * xmlParser,
            IXML_Node * node )  [static]
```

Processes the CONTENT as defined in XML spec.

Parameters

in	*xmlParser*	The XML parser.
in	*node*	The Node to process.

References _Parser::curPtr, IxmlPrintf(), Parser_clearTokenBuf(), Parser_copyToken(), Parser_processCDSect(), Parser_skipComment(), Parser_skipPI(), Parser_skipWhiteSpaces(), safe_strdup(), and _Parser::savePtr.

Referenced by Parser_getNextNode().

9.15.2.33 Parser_processElementName()

```
static int Parser_processElementName (
            IXML_Document * rootDoc,
            Parser * xmlParser,
            IXML_Node * newNode )  [static]
```

Processes element name.

Returns

IXML_SUCCESS if successful, otherwise or an error code.

Parameters

in	*rootDoc*	The XML document.
in	*xmlParser*	The XML parser.
in	*newNode*	The Node to process.

References _Parser::curPtr, isTopLevelElement(), ixmlDocument_createElementEx(), ixmlElement_free(), ixmlNode_appendChild(), ixmlNode_setNodeProperties(), Parser_ElementPrefixDefined(), Parser_hasDefaultNamespace(), Parser_pushElement(), Parser_setElementNamespace(), and _Parser::savePtr.

Referenced by Parser_parseDocument().

9.15.2.34 Parser_processETag()

```
static int Parser_processETag (
            Parser * xmlParser,
            IXML_Node * node,
            int * bETag )  [static]
```

Process ETag as defined by XML spec.

Parameters

in	*xmlParser*	The XML parser.
in	*node*	The Node to process.
out	*bETag*	.

References IxmlPrintf(), Parser_getNextToken(), Parser_skipWhiteSpaces(), and safe_strdup().

Referenced by Parser_getNextNode().

9.15.2.35 Parser_processSTag()

```
static int Parser_processSTag (
            Parser * xmlParser,
            IXML_Node * node )  [static]
```

Processes the STag as defined by XML spec.

Parameters

in	*xmlParser*	The XML parser.
in	*node*	The Node to process.

References _Parser::curPtr, Parser_getNextToken(), Parser_setLastElem(), Parser_setNodePrefixAndLocalName(), safe_strdup(), and _Parser::savePtr.

Referenced by Parser_getNextNode().

9.15.2.36 Parser_pushElement()

```
static int Parser_pushElement (
            Parser * xmlParser,
            IXML_Node * newElement )  [static]
```

Push a new element onto element stack.

Returns

Parameters

in	*xmlParser*	The XML parser.
in	*newElement*	The element node to push.

References Parser_freeElementStackItem(), and safe_strdup().

Referenced by Parser_processElementName().

9.15.2.37 Parser_readFileOrBuffer()

```
static int Parser_readFileOrBuffer (
            Parser * xmlParser,
            const char * xmlFileName,
            int file )  [static]
```

Read a xml file or buffer contents into xml parser.

Parameters

in	*xmlParser*	The XML parser.
in	*xmlFileName*	The file name or the buffer to copy, according to the parameter "file".
in	*file*	1 if you want to read from a file, 0 if xmlFileName is the buffer to copy to the parser.

References _Parser::dataBuffer, and safe_strdup().

Referenced by Parser_LoadDocument().

9.15.2.38 Parser_setElementNamespace()

```
static int Parser_setElementNamespace (
            IXML_Element * newElement,
            const char * nsURI )  [static]
```

Set element's namespace.

Parameters

in	*newElement*	The Element Node to process.
in	*nsURI*	The name space string.

References safe_strdup().

Referenced by Parser_processElementName().

9.15.2.39 Parser_setErrorChar()

```
void Parser_setErrorChar (
            char c )
```

Sets the error character.

If 'c' is 0 (default), the parser is strict about XML encoding: invalid UTF-8 sequences or "&" entities are rejected, and the parsing aborts.

If 'c' is not 0, the parser is relaxed: invalid UTF-8 characters are replaced by this character, and invalid "&" entities are left untranslated. The parsing is then allowed to continue.

Parameters

in	*c*	The character to become the error character.

Referenced by ixmlRelaxParser().

9.15.2.40 Parser_setLastElem()

```
static int Parser_setLastElem (
            Parser * xmlParser,
            const char * s )  [static]
```

Set the last element to be the given string.

Parameters

in	*xmlParser*	The XML parser.
in	*s*	The string to copy from.

References ixml_membuf_assign_str().

Referenced by Parser_processSTag().

9.15.2.41 Parser_setNodePrefixAndLocalName()

```
int Parser_setNodePrefixAndLocalName (
            IXML_Node * node )
```

Set the node prefix and localName as defined by the nodeName in the form of ns:name.

Parameters

in,out	*node*	The Node to process.

References safe_strdup().

Referenced by ixmlNode_setNodeName(), Parser_processSTag(), and Parser_xmlNamespace().

9.15.2.42 Parser_skipBom()

```
static void Parser_skipBom (
            Parser * xmlParser )  [static]
```

Skip UTF-8 byte order mark.

Parameters

in	*xmlParser*	The XML parser.

References _Parser::curPtr.

Referenced by Parser_skipProlog().

9.15.2.43 Parser_skipComment()

```
static int Parser_skipComment (
            char ** pstrSrc )  [static]
```

Skips all characters in the string until it finds the skip key. Then it skips the skip key and returns.

Parameters

in,out	*pstrSrc*	The pointer to the skipped point.

Referenced by Parser_processContent(), and Parser_skipMisc().

9.15.2.44 Parser_skipDocType()

```
static int Parser_skipDocType (
            char ** pstr )  [static]
```

Skips document type declaration.

Parameters

in,out	*pstr*	The pointer to the skipped point.

Referenced by Parser_skipProlog().

9.15.2.45 Parser_skipMisc()

```
static int Parser_skipMisc (
            Parser * xmlParser )  [static]
```

Skip comment, PI and white space.

Parameters

in	*xmlParser*	The XML parser.

References _Parser::curPtr, Parser_skipComment(), Parser_skipString(), and Parser_skipWhiteSpaces().

Referenced by Parser_getNextToken(), and Parser_skipProlog().

9.15.2.46 Parser_skipPI()

```
static int Parser_skipPI (
            char ** pSrc )  [static]
```

Parameters

in,out	*pSrc*	The pointer to the skipped point.

Referenced by Parser_processContent().

9.15.2.47 Parser_skipProlog()

```
static int Parser_skipProlog (
            Parser * xmlParser )  [static]
```

Skip prolog.

Parameters

in,out	*xmlParser*	The XML parser.

References _Parser::curPtr, Parser_skipBom(), Parser_skipDocType(), Parser_skipMisc(), Parser_skipWhiteSpaces(), and Parser_skipXMLDecl().

Referenced by Parser_parseDocument().

9.15.2.48 Parser_skipString()

```
static int Parser_skipString (
            char ** pstrSrc,
            const char * strSkipKey )  [static]
```

Skips all characters in the string until it finds the skip key. Then it skips the skip key and returns.

Parameters

in,out	*pstrSrc*	The pointer to the skipped point.
in	*strSkipKey*	The skip key.

Referenced by Parser_skipMisc(), and Parser_skipXMLDecl().

9.15.2.49 Parser_skipWhiteSpaces()

```
static void Parser_skipWhiteSpaces (
            Parser * xmlParser )  [static]
```

Skip white spaces.

Parameters

in	*xmlParser*	The XML parser.

References _Parser::curPtr.

Referenced by Parser_getNextNode(), Parser_processContent(), Parser_processETag(), Parser_skipMisc(), Parser_skipProlog(), and Parser_skipXMLDecl().

9.15.2.50 Parser_skipXMLDecl()

```
static int Parser_skipXMLDecl (
            Parser * xmlParser )  [static]
```

Skips XML declarations.

Parameters

in,out	*xmlParser*	The XML parser.

References _Parser::curPtr, Parser_skipString(), and Parser_skipWhiteSpaces().

Referenced by Parser_skipProlog().

9.15.2.51 Parser_UTF8ToInt()

```
static int Parser_UTF8ToInt (
            const char * ss,
            ptrdiff_t * len )  [static]
```

In UTF-8, characters are encoded using sequences of 1 to 6 octets. This functions will return a UTF-8 character value and its octets number.

Returns

The UTF-8 character converted to an int (32 bits).

Parameters

in	*ss*	The pointer to the character to encode.
out	*len*	The number of octets of the UTF-8 encoding of this character.

References IxmlPrintf().

Referenced by Parser_getChar(), Parser_getNextToken(), and Parser_processAttribute().

9.15.2.52 Parser_xmlNamespace()

```
static int Parser_xmlNamespace (
            Parser * xmlParser,
            IXML_Node * newNode )  [static]
```

Add namespace definition.

Parameters

in	*xmlParser*	The XML parser.
in	*newNode*	The Node to process.

References IxmlPrintf(), Parser_addNamespace(), Parser_freeNsURI(), Parser_setNodePrefixAndLocalName(), and safe_strdup().

Referenced by Parser_processAttribute().

9.15.2.53 safe_strdup()

```
static char * safe_strdup (
            const char * s )  [static]
```

Version of strdup() that handles NULL input.

Returns

The same as strdup().

Parameters

in	*s*	String to be duplicated.

Referenced by Parser_addNamespace(), Parser_getNextNode(), Parser_processAttribute(), Parser_processCDSect(), Parser_processContent(), Parser_processETag(), Parser_processSTag(), Parser_pushElement(), Parser_readFileOrBuffer(), Parser_setElementNamespace(), Parser_setNodePrefixAndLocalName(), and Parser_xmlNamespace().

9.15.3 Variable Documentation

9.15.3.1 Letter

```
char_info_t Letter[]  [static]
```

The letter table contains all characters in XML 1.0 plus ":", "_" and ideographic.

This table contains all the characters that an element name can start with. See XML 1.0 (2nd Edition) for more details.

Referenced by Parser_isNameChar().

9.15.3.2 NameChar

```
char_info_t NameChar[]  [static]
```

The NameChar table contains CombiningChar, Extender, Digit, '-', '.', less '_', ':'.

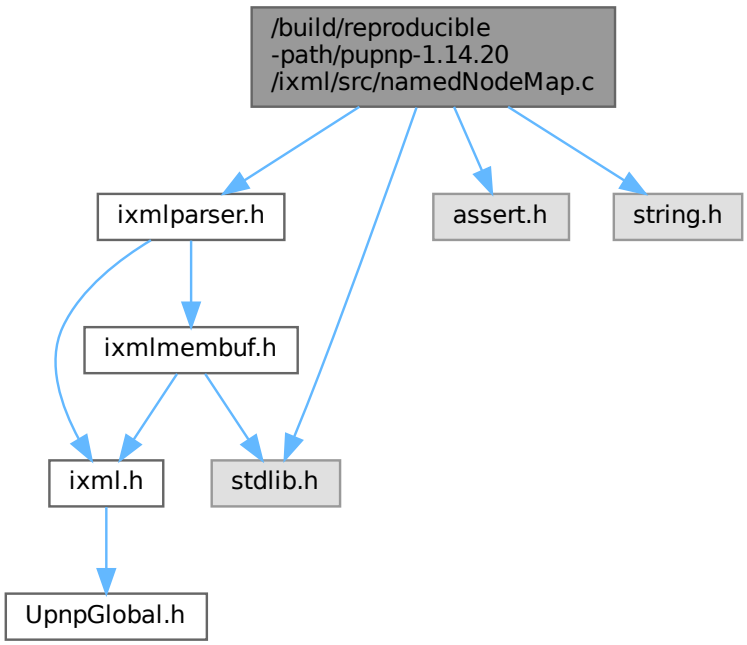
NameChar ::= Digit | '-' | '.' | CombiningChar | Extender See XML 1.0 2nd Edition

Referenced by Parser_isNameChar().

9.16 /build/reproducible-path/pupnp-1.14.20/ixml/src/namedNodeMap.c File Reference

```
#include "ixmlparser.h"
#include <assert.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for namedNodeMap.c:



Functions

- static unsigned long ixmlNamedNodeMap_getItemNumber (IXML_NamedNodeMap *nnMap, const char *name)

  Return the item number of a item in NamedNodeMap.
- void ixmlNamedNodeMap_init (IXML_NamedNodeMap *nnMap)

  Initializes a NamedNodeMap object.
- IXML_Node * **ixmlNamedNodeMap_getNamedItem** (IXML_NamedNodeMap *nnMap, const DOMString name)
- IXML_Node * ixmlNamedNodeMap_item (IXML_NamedNodeMap *nnMap, unsigned long index)

  *Retrieves the indexth item in the map. If index is greater than or equal to the number of nodes in this map, this returns **NULL**.*
- unsigned long ixmlNamedNodeMap_getLength (IXML_NamedNodeMap *nnMap)

  *Returns the number of items contained in this **NamedNodeMap**.*
- void ixmlNamedNodeMap_free (IXML_NamedNodeMap *nnMap)

  *Frees a **NamedNodeMap**.*
- int ixmlNamedNodeMap_addToNamedNodeMap (IXML_NamedNodeMap **nnMap, IXML_Node *add)

  Add a node to a NamedNodeMap.

9.16.1 Function Documentation

9.16.1.1 ixmlNamedNodeMap_addToNamedNodeMap()

```
int ixmlNamedNodeMap_addToNamedNodeMap (
            IXML_NamedNodeMap ** nnMap,
            IXML_Node * add )
```

Add a node to a NamedNodeMap.

Returns

IXML_SUCCESS or failure.

References ixmlNamedNodeMap_init().

Referenced by ixmlNode_getAttributes().

9.16.1.2 ixmlNamedNodeMap_getItemNumber()

```
static unsigned long ixmlNamedNodeMap_getItemNumber (
            IXML_NamedNodeMap * nnMap,
            const char * name )  [static]
```

Return the item number of a item in NamedNodeMap.

Parameters

in	*nnMap*	The named node map to process.
in	*name*	The name of the item to find.

9.16.1.3 ixmlNamedNodeMap_init()

```
void ixmlNamedNodeMap_init (
                IXML_NamedNodeMap * nnMap )
```

Initializes a NamedNodeMap object.

Parameters

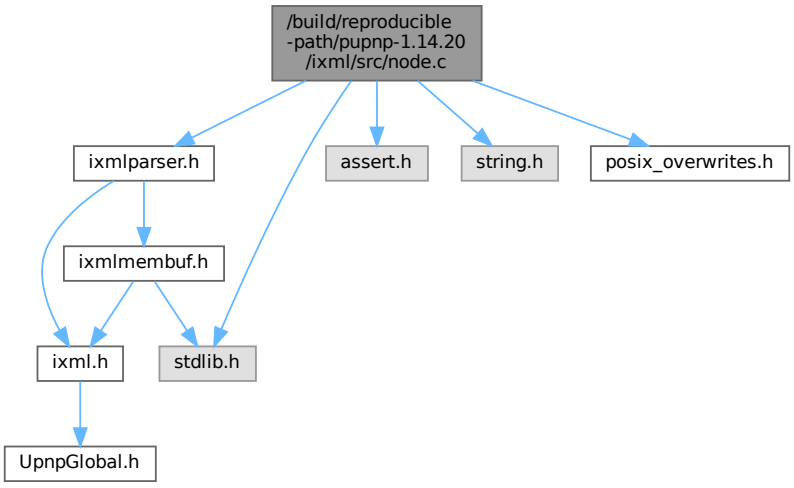
in	*nnMap*	The named node map to process.

Referenced by ixmlNamedNodeMap_addToNamedNodeMap(), and ixmlNode_getAttributes().

9.17 /build/reproducible-path/pupnp-1.14.20/ixml/src/node.c File Reference

```
#include "ixmlparser.h"
#include <assert.h>
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for node.c:



Functions

- void ixmlNode_init (IXML_Node *nodeptr)

  Intializes a node.
- void ixmlCDATASection_init (IXML_CDATASection *nodeptr)

  *Initializes a **CDATASection** node.*
- void ixmlCDATASection_free (IXML_CDATASection *nodeptr)

*Frees a **CDATASection** node.*

- static void ixmlNode_freeSingleNode (IXML_Node *nodeptr)

  Frees a node content.
- void ixmlNode_free (IXML_Node *nodeptr)

  *Frees a **Node** and all **Nodes** in its subtree.*
- const DOMString ixmlNode_getNodeName (IXML_Node *nodeptr)

  *Returns the name of the **Node**, depending on what type of **Node** it is, in a read-only string.*
- const DOMString ixmlNode_getLocalName (IXML_Node *nodeptr)

  *Retrieves the local name of a **Node**, if present.*
- static int ixmlNode_setNamespaceURI (IXML_Node *nodeptr, const char *namespaceURI)

  Sets the namespace URI of the node.
- static int ixmlNode_setPrefix (IXML_Node *nodeptr, const char *prefix)
- static int ixmlNode_setLocalName (IXML_Node *nodeptr, const char *localName)

  Set the localName of the node.
- const DOMString ixmlNode_getNamespaceURI (IXML_Node *nodeptr)

  *Retrieves the namespace URI for a **Node** as a **DOMString**.*
- const DOMString ixmlNode_getPrefix (IXML_Node *nodeptr)

  Retrieves the namespace prefix, if present.
- const DOMString ixmlNode_getNodeValue (IXML_Node *nodeptr)

  *Returns the value of the **Node** as a string.*
- int ixmlNode_setNodeValue (IXML_Node *nodeptr, const char *newNodeValue)

  *Assigns a new value to a **Node**.*
- unsigned short ixmlNode_getNodeType (IXML_Node *nodeptr)

  *Retrieves the type of a **Node**. Note that not all possible return values are actually implemented.*
- IXML_Node * ixmlNode_getParentNode (IXML_Node *nodeptr)

  *Retrieves the parent **Node** for a **Node**.*
- IXML_Node * ixmlNode_getFirstChild (IXML_Node *nodeptr)

  *Retrieves the first child **Node** of a **Node**.*
- IXML_Node * ixmlNode_getLastChild (IXML_Node *nodeptr)

  *Retrieves the last child **Node** of a **Node**.*
- IXML_Node * ixmlNode_getPreviousSibling (IXML_Node *nodeptr)

  *Retrieves the sibling **Node** immediately preceding this **Node**.*
- IXML_Node * ixmlNode_getNextSibling (IXML_Node *nodeptr)

  *Retrieves the sibling **Node** immediately following this **Node**.*
- IXML_Document * ixmlNode_getOwnerDocument (IXML_Node *nodeptr)

  *Retrieves the document object associated with this **Node**.*
- static int ixmlNode_isAncestor (IXML_Node *ancestorNode, IXML_Node *toFind)

  Check if ancestorNode is ancestor of toFind.
- static int ixmlNode_isParent (IXML_Node *nodeptr, IXML_Node *toFind)

  Check whether toFind is a children of nodeptr.
- static int ixmlNode_allowChildren (IXML_Node *nodeptr, IXML_Node *newChild)

  Check to see whether nodeptr allows children of type newChild.
- int ixmlNode_compare (IXML_Node *srcNode, IXML_Node *destNode)

  Compare two nodes to see whether they are the same node. Parent, sibling and children node are ignored.
- int ixmlNode_insertBefore (IXML_Node *nodeptr, IXML_Node *newChild, IXML_Node *refChild)

  *Inserts a new child **Node** before the existing child **Node**.*
- int ixmlNode_replaceChild (IXML_Node *nodeptr, IXML_Node *newChild, IXML_Node *oldChild, IXML_Node **returnNode)

  *Replaces an existing child **Node** with a new child **Node** in the list of children of a **Node**.*
- int ixmlNode_removeChild (IXML_Node *nodeptr, IXML_Node *oldChild, IXML_Node **returnNode)

  *Removes a child from the list of children of a **Node**.*

- int ixmlNode_appendChild (IXML_Node *nodeptr, IXML_Node *newChild)

  *Appends a child **Node** to the list of children of a **Node**.*
- static IXML_Node * ixmlNode_cloneTextNode (IXML_Node *nodeptr)

  Returns a clone of nodeptr.
- static IXML_CDATASection * ixmlNode_cloneCDATASect (IXML_CDATASection *nodeptr)

  Return a clone of CDATASection node.
- static IXML_Element * ixmlNode_cloneElement (IXML_Element *nodeptr)

  Returns a clone of element node.
- static IXML_Document * ixmlNode_newDoc (void)

  Returns a new document node.
- static IXML_Attr * ixmlNode_cloneAttr (IXML_Attr *nodeptr)

  Returns a clone of an attribute node.
- static IXML_Attr * ixmlNode_cloneAttrDirect (IXML_Attr *nodeptr)

  Return a clone of attribute node, with specified field set to 1.
- static void ixmlNode_setSiblingNodesParent (IXML_Node *nodeptr)

  Sets siblings nodes parent to be the same as this node's.
- static IXML_Node * ixmlNode_cloneNodeTreeRecursive (IXML_Node *nodeptr, int deep)

  Recursive function that clones a node tree of nodeptr.
- static IXML_Node * ixmlNode_cloneNodeTree (IXML_Node *nodeptr, int deep)

  Function that clones a node tree of nodeptr.
- IXML_Node * ixmlNode_cloneNode (IXML_Node *nodeptr, int deep)

  *Clones a **Node**.*
- IXML_NodeList * ixmlNode_getChildNodes (IXML_Node *nodeptr)

  *Retrieves the list of children of a **Node** in a **NodeList** structure.*
- IXML_NamedNodeMap * ixmlNode_getAttributes (IXML_Node *nodeptr)

  *Retrieves the attributes of a **Node**, if it is an **Element** node, in a **NamedNodeMap** structure.*
- int ixmlNode_hasChildNodes (IXML_Node *nodeptr)

  *Queries whether or not a **Node** has children.*
- int ixmlNode_hasAttributes (IXML_Node *nodeptr)

  *Queries whether this **Node** has attributes.*
- static void ixmlNode_getElementsByTagNameRecursive (IXML_Node *n, const char *tagname, IXML_NodeList **list)

  Recursively traverse the whole tree, search for element with the given tagname.
- void ixmlNode_getElementsByTagName (IXML_Node *n, const char *tagname, IXML_NodeList **list)

  Returns a nodeList of all descendant Elements with a given tagName, in the order in which they are encountered in a traversal of this element tree.
- static void ixmlNode_getElementsByTagNameNSRecursive (IXML_Node *n, const char *namespaceURI, const char *localName, IXML_NodeList **list)
- void ixmlNode_getElementsByTagNameNS (IXML_Node *n, const char *namespaceURI, const char *localName, IXML_NodeList **list)

  Returns a nodeList of all the descendant Elements with a given local name and namespace URI in the order in which they are encountered in a preorder traversal of this Elememt tree.
- int ixmlNode_setNodeName (IXML_Node *node, const DOMString qualifiedName)
- int ixmlNode_setNodeProperties (IXML_Node *destNode, IXML_Node *src)

9.17.1 Function Documentation

9.17.1.1 ixmlNode_allowChildren()

```
static int ixmlNode_allowChildren (
            IXML_Node * nodeptr,
            IXML_Node * newChild )  [static]
```

Check to see whether nodeptr allows children of type newChild.

Returns

- 1, if nodeptr can have newChild as children.
- 0, if nodeptr cannot have newChild as children.

Parameters

in	*nodeptr*	The **Node** to check.
in	*newChild*	The child **Node** to check.

Referenced by ixmlNode_appendChild(), ixmlNode_insertBefore(), and ixmlNode_replaceChild().

9.17.1.2 ixmlNode_cloneAttr()

```
static IXML_Attr * ixmlNode_cloneAttr (
            IXML_Attr * nodeptr )  [static]
```

Returns a clone of an attribute node.

Returns

A clone of an attribute node.

Parameters

in	*nodeptr*	The **Node** to clone.

References ixmlAttr_free(), ixmlNode_setLocalName(), ixmlNode_setNamespaceURI(), ixmlNode_setNodeName(), ixmlNode_setNodeValue(), and ixmlNode_setPrefix().

Referenced by ixmlNode_cloneAttrDirect(), and ixmlNode_cloneNodeTreeRecursive().

9.17.1.3 ixmlNode_cloneAttrDirect()

```
static IXML_Attr * ixmlNode_cloneAttrDirect (
            IXML_Attr * nodeptr )  [static]
```

Return a clone of attribute node, with specified field set to 1.

Returns

A clone of attribute node, with specified field set to 1.

Parameters

in	*nodeptr*	The **Node** to clone.

References ixmlNode_cloneAttr().

Referenced by ixmlNode_cloneNode().

9.17.1.4 ixmlNode_cloneCDATASect()

```
static IXML_CDATASection * ixmlNode_cloneCDATASect (
            IXML_CDATASection * nodeptr )  [static]
```

Return a clone of CDATASection node.

Returns

A clone of CDATASection node.

Parameters

in	*nodeptr*	The **Node** to clone.

References ixmlCDATASection_free(), ixmlCDATASection_init(), ixmlNode_setNodeName(), and ixmlNode_setNodeValue().

Referenced by ixmlNode_cloneNodeTreeRecursive().

9.17.1.5 ixmlNode_cloneElement()

```
static IXML_Element * ixmlNode_cloneElement (
            IXML_Element * nodeptr )  [static]
```

Returns a clone of element node.

Returns

A clone of element node.

Parameters

in	*nodeptr*	The **Node** to clone.

References ixmlElement_free(), ixmlElement_init(), ixmlElement_setTagName(), ixmlNode_setLocalName(), ixmlNode_setNamespaceURI(), ixmlNode_setNodeName(), ixmlNode_setNodeValue(), and ixmlNode_setPrefix().

Referenced by ixmlNode_cloneNodeTree(), and ixmlNode_cloneNodeTreeRecursive().

9.17.1.6 ixmlNode_cloneNodeTree()

```
static IXML_Node * ixmlNode_cloneNodeTree (
            IXML_Node * nodeptr,
            int deep )  [static]
```

Function that clones a node tree of nodeptr.

Returns

The cloned node/tree.

Parameters

in	*nodeptr*	Node tree to clone.
in	*deep*	1 if you want to clone the tree.

References ixmlNode_cloneElement(), and ixmlNode_cloneNodeTreeRecursive().

Referenced by ixmlNode_cloneNode().

9.17.1.7 ixmlNode_cloneNodeTreeRecursive()

```
static IXML_Node * ixmlNode_cloneNodeTreeRecursive (
            IXML_Node * nodeptr,
            int deep )  [static]
```

Recursive function that clones a node tree of nodeptr.

Returns

The cloned node/tree.

Parameters

in	*nodeptr*	Node tree to clone.
in	*deep*	1 if you want to clone the tree.

References ixmlNode_cloneAttr(), ixmlNode_cloneCDATASect(), ixmlNode_cloneElement(), ixmlNode_cloneNodeTreeRecursive(), ixmlNode_cloneTextNode(), ixmlNode_newDoc(), and ixmlNode_setSiblingNodesParent().

Referenced by ixmlNode_cloneNodeTree(), and ixmlNode_cloneNodeTreeRecursive().

9.17.1.8 ixmlNode_cloneTextNode()

```
static IXML_Node * ixmlNode_cloneTextNode (
            IXML_Node * nodeptr )  [static]
```

Returns a clone of nodeptr.

Returns

A cloned node of nodeptr.

Parameters

in	*nodeptr*	The **Node** to clone.

References ixmlNode_free(), ixmlNode_init(), ixmlNode_setNodeName(), and ixmlNode_setNodeValue().

Referenced by ixmlNode_cloneNodeTreeRecursive().

9.17.1.9 ixmlNode_compare()

```
int ixmlNode_compare (
            IXML_Node * srcNode,
            IXML_Node * destNode )
```

Compare two nodes to see whether they are the same node. Parent, sibling and children node are ignored.

Returns

- 1, the two nodes are the same.
- 0, the two nodes are not the same.

Parameters

in	*srcNode*	The first **Node**.
in	*destNode*	The second **Node**.

Referenced by ixmlElement_findAttributeNode().

9.17.1.10 ixmlNode_freeSingleNode()

```
static void ixmlNode_freeSingleNode (
            IXML_Node * nodeptr )  [static]
```

Frees a node content.

Parameters

in	*nodeptr*	The node to free.

Referenced by ixmlNode_free().

9.17.1.11 ixmlNode_getElementsByTagName()

```
void ixmlNode_getElementsByTagName (
            IXML_Node * n,
            const char * tagname,
            IXML_NodeList ** list )
```

Returns a nodeList of all descendant Elements with a given tagName, in the order in which they are encountered in a traversal of this element tree.

Parameters

in	*n*	The **Node** tree.
in	*tagname*	The tag name to match.
out	*list*	The output **NodeList**.

References ixmlNode_getElementsByTagNameRecursive(), ixmlNode_getFirstChild(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), and ixmlNodeList_addToNodeList().

9.17.1.12 ixmlNode_getElementsByTagNameNS()

```
void ixmlNode_getElementsByTagNameNS (
            IXML_Node * n,
            const char * namespaceURI,
            const char * localName,
            IXML_NodeList ** list )
```

Returns a nodeList of all the descendant Elements with a given local name and namespace URI in the order in which they are encountered in a preorder traversal of this Elememt tree.

Parameters

in	*n*	The **Element** tree.
in	*namespaceURI*	The name space to match.
in	*localName*	The local name to match.
out	*list*	The output **NodeList**.

References DOMString, ixmlNode_getElementsByTagNameNSRecursive(), ixmlNode_getFirstChild(), ixmlNode_getLocalName(), ixmlNode_getNamespaceURI(), ixmlNode_getNodeType(), and ixmlNodeList_addToNodeList().

9.17.1.13 ixmlNode_getElementsByTagNameNSRecursive()

```
static void ixmlNode_getElementsByTagNameNSRecursive (
            IXML_Node * n,
            const char * namespaceURI,
            const char * localName,
            IXML_NodeList ** list )  [static]
```

Parameters

in	*n*	.
in	*namespaceURI*	.
in	*localName*	.
out	*list*	.

References DOMString, ixmlNode_getElementsByTagNameNSRecursive(), ixmlNode_getFirstChild(), ixmlNode_getLocalName(), ixmlNode_getNamespaceURI(), ixmlNode_getNextSibling(), ixmlNode_getNodeType(), and ixmlNodeList_addToNodeList().

Referenced by ixmlNode_getElementsByTagNameNS(), and ixmlNode_getElementsByTagNameNSRecursive().

9.17.1.14 ixmlNode_getElementsByTagNameRecursive()

```
static void ixmlNode_getElementsByTagNameRecursive (
            IXML_Node * n,
            const char * tagname,
            IXML_NodeList ** list )  [static]
```

Recursively traverse the whole tree, search for element with the given tagname.

Parameters

in	*n*	The **Node** tree.
in	*tagname*	The tag name to match.
out	*list*	The output **NodeList**.

References ixmlNode_getElementsByTagNameRecursive(), ixmlNode_getFirstChild(), ixmlNode_getNextSibling(), ixmlNode_getNodeName(), ixmlNode_getNodeType(), and ixmlNodeList_addToNodeList().

Referenced by ixmlNode_getElementsByTagName(), and ixmlNode_getElementsByTagNameRecursive().

9.17.1.15 ixmlNode_init()

```
void ixmlNode_init (
            IXML_Node * nodeptr )
```

Intializes a node.

Parameters

in	*nodeptr*	The **Node** to iniatialize.

Referenced by ixmlNode_cloneTextNode(), and Parser_parseDocument().

9.17.1.16 ixmlNode_isAncestor()

```
static int ixmlNode_isAncestor (
            IXML_Node * ancestorNode,
            IXML_Node * toFind )  [static]
```

Check if ancestorNode is ancestor of toFind.

Returns

1 or 0.

Parameters

in	*ancestorNode*	The candidate to ancestor **Node**.
in	*toFind*	The **Node** to check for an ancestor.

References ixmlNode_isAncestor().

Referenced by ixmlNode_appendChild(), ixmlNode_insertBefore(), ixmlNode_isAncestor(), and ixmlNode_replaceChild().

9.17.1.17 ixmlNode_isParent()

```
static int ixmlNode_isParent (
            IXML_Node * nodeptr,
            IXML_Node * toFind )  [static]
```

Check whether toFind is a children of nodeptr.

Returns

1 or 0.

Parameters

in	*nodeptr*	The candidate to parent **Node**.
in	*toFind*	The **Node** to check for his parent.

Referenced by ixmlNode_appendChild(), ixmlNode_insertBefore(), ixmlNode_removeChild(), and ixmlNode_replaceChild().

9.17.1.18 ixmlNode_newDoc()

```
static IXML_Document * ixmlNode_newDoc (
            void  )  [static]
```

Returns a new document node.

Currently, the IXML_Document struct is just a node, so this function just mallocs the IXML_Document, sets the node type and name.

Returns

A new document node.

References ixmlDocument_free(), ixmlDocument_init(), and ixmlNode_setNodeName().

Referenced by ixmlNode_cloneNodeTreeRecursive().

9.17.1.19 ixmlNode_setLocalName()

```
static int ixmlNode_setLocalName (
            IXML_Node * nodeptr,
            const char * localName )  [static]
```

Set the localName of the node.

Returns

IXML_SUCCESS or failure.

Parameters

in	*nodeptr*	The pointer to the node.
in	*localName*	The local name to set.

Referenced by ixmlNode_cloneAttr(), ixmlNode_cloneElement(), and ixmlNode_setNodeProperties().

9.17.1.20 ixmlNode_setNamespaceURI()

```
static int ixmlNode_setNamespaceURI (
            IXML_Node * nodeptr,
            const char * namespaceURI )  [static]
```

Sets the namespace URI of the node.

Parameters

in	*nodeptr*	The **Node** on which to operate.
in	*namespaceURI*	The name space string to set.

Referenced by ixmlNode_cloneAttr(), and ixmlNode_cloneElement().

9.17.1.21 ixmlNode_setNodeName()

```
int ixmlNode_setNodeName (
            IXML_Node * node,
            const DOMString qualifiedName )
```

Returns

Parameters

in	*node*	The **Node**.
in	*qualifiedName*	.

References Parser_setNodePrefixAndLocalName().

Referenced by ixmlNode_cloneAttr(), ixmlNode_cloneCDATASect(), ixmlNode_cloneElement(), ixmlNode_cloneTextNode(), and ixmlNode_newDoc().

9.17.1.22 ixmlNode_setNodeProperties()

```
int ixmlNode_setNodeProperties (
            IXML_Node * destNode,
            IXML_Node * src )
```

Returns

Parameters

in	*destNode*	.
in	*src*	.

References ixmlNode_setLocalName(), ixmlNode_setNodeValue(), and ixmlNode_setPrefix().

Referenced by Parser_processAttributeName(), and Parser_processElementName().

9.17.1.23 ixmlNode_setPrefix()

```
static int ixmlNode_setPrefix (
            IXML_Node * nodeptr,
            const char * prefix )  [static]
```

Parameters

in	*nodeptr*	The **Node** on which to operate.
in	*prefix*	The prefix string to set.

Referenced by ixmlNode_cloneAttr(), ixmlNode_cloneElement(), and ixmlNode_setNodeProperties().

9.17.1.24 ixmlNode_setSiblingNodesParent()

```
static void ixmlNode_setSiblingNodesParent (
            IXML_Node * nodeptr )  [static]
```

Sets siblings nodes parent to be the same as this node's.

Parameters

in	*nodeptr*	The node to operate on.

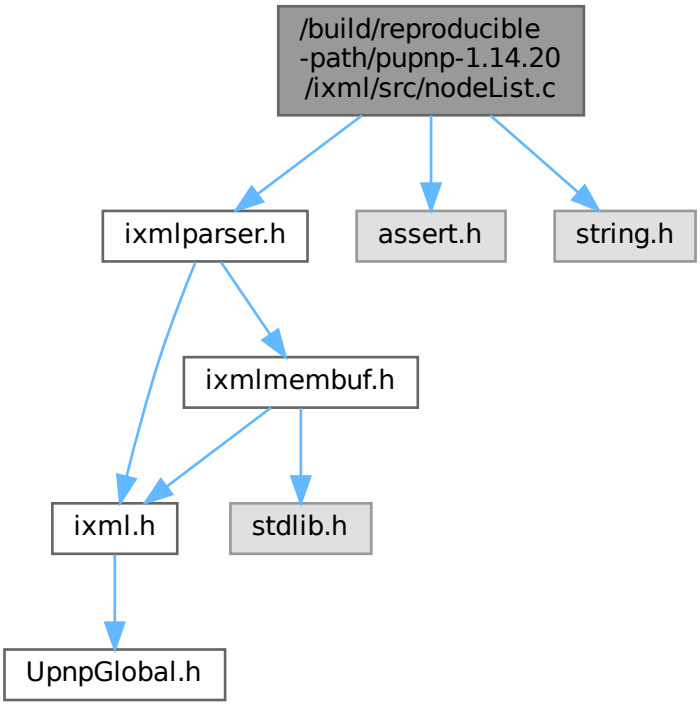
Referenced by ixmlNode_cloneNodeTreeRecursive().

9.18 /build/reproducible-path/pupnp-1.14.20/ixml/src/nodeList.c File Reference

```
#include "ixmlparser.h"
#include <assert.h>
```

```
#include <string.h>
```

Include dependency graph for nodeList.c:



Functions

- void ixmlNodeList_init (IXML_NodeList *nList)

  Initializes a nodelist.
- IXML_Node * ixmlNodeList_item (IXML_NodeList *nList, unsigned long index)

  *Retrieves a **Node** from a **NodeList** specified by a numerical index.*
- int ixmlNodeList_addToNodeList (IXML_NodeList **nList, IXML_Node *add)

  Add a node to nodelist.
- unsigned long ixmlNodeList_length (IXML_NodeList *nList)

  *Returns the number of **Nodes** in a **NodeList**.*
- void ixmlNodeList_free (IXML_NodeList *nList)

  *Frees a **NodeList** object.*

9.18.1 Function Documentation

9.18.1.1 ixmlNodeList_addToNodeList()

```
int ixmlNodeList_addToNodeList (
            IXML_NodeList ** nList,
            IXML_Node * add )
```

Add a node to nodelist.

Parameters

in	*nList*	The pointer to the nodelist.
in	*add*	The node to add.

References ixmlNodeList_init().

Referenced by ixmlNode_getChildNodes(), ixmlNode_getElementsByTagName(), ixmlNode_getElementsByTagNameNS(), ixmlNode_getElementsByTagNameNSRecursive(), and ixmlNode_getElementsByTagNameRecursive().

9.18.1.2 ixmlNodeList_init()

```
void ixmlNodeList_init (
             IXML_NodeList * nList )
```

Initializes a nodelist.

Parameters

in,out	*nList*	The **NodeList** to initialize.

Referenced by ixmlNode_getChildNodes(), and ixmlNodeList_addToNodeList().

9.19 generator.h

```
00001 #ifndef GENERATOR_H
00002 #define GENERATOR_H
00003
00004 #define ARRAY_SIZE(a) (sizeof(a) / sizeof 0 [a])
00005
00006 enum e_Type
00007 {
00008         TYPE_CLASS = 0,
00009         TYPE_INTEGER,
00010         TYPE_BUFFER,
00011         TYPE_LIST,
00012         TYPE_STRING,
00013         TYPE_DOMSTRING
00014 };
00015
00016 struct s_Type_Integer
00017 {
00018         const char *int_type;
00019 };
00020
00021 struct s_Member
00022 {
00023         const char *name;
00024         enum e_Type type;
00025         const char *type_name;
00026         const char *header;
00027 };
00028
00029 /* clang-format off */
00030 #define INIT_MEMBER(NAME, TYPE, TYPE_NAME, HEADER) \
00031         { \
00032                 .name = #NAME, \
00033                 .type = TYPE, \
00034                 .type_name = #TYPE_NAME, \
00035                 .header = HEADER \
00036         }
00037 /* clang-format on */
00038
00039 struct s_Class
00040 {
```

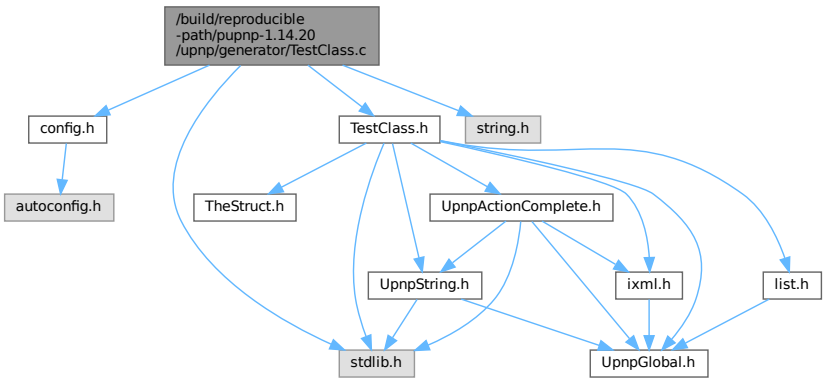
```
00041         const char *name;
00042         unsigned n_members;
00043         struct s_Member *members;
00044         const char *source;
00045         const char *header;
00046 };
00047
00048 /* clang-format off */
00049 #define INIT_CLASS(CLASS) \
00050         { \
00051                 .name = #CLASS, \
00052                 .n_members = ARRAY_SIZE(CLASS##_members), \
00053                 .members = CLASS##_members, \
00054                 .source = #CLASS ".c", \
00055                 .header = #CLASS ".h", \
00056         }
00057 /* clang-format on */
00058
00059 #endif /* GENERATOR_H */
```

9.20 /build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.c File Reference

Source file for TestClass methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "TestClass.h"
```

Include dependency graph for TestClass.c:



Data Structures

- struct s_TestClass

Functions

- TestClass * TestClass_new ()
- void TestClass_delete (TestClass *q)
- int TestClass_assign (TestClass *p, const TestClass *q)
- TestClass * TestClass_dup (const TestClass *q)

- const UpnpListHead * TestClass_get_TheList (const TestClass *p)
- int TestClass_set_TheList (TestClass *p, const UpnpListHead *q)
- void TestClass_add_to_list_TheList (TestClass *p, struct UpnpListHead *head)
- int TestClass_get_TheInteger (const TestClass *p)
- int TestClass_set_TheInteger (TestClass *p, int n)
- unsigned long TestClass_get_TheUnsignedLong (const TestClass *p)
- int TestClass_set_TheUnsignedLong (TestClass *p, unsigned long n)
- char * TestClass_get_TheCharPointer (const TestClass *p)
- int TestClass_set_TheCharPointer (TestClass *p, char *n)
- const struct TheStruct * TestClass_get_TheBuffer (const TestClass *p)
- int TestClass_set_TheBuffer (TestClass *p, const struct TheStruct *buf)
- void TestClass_clear_TheBuffer (TestClass *p)
- const UpnpActionComplete * TestClass_get_TheActionComplete (const TestClass *p)
- int TestClass_set_TheActionComplete (TestClass *p, const UpnpActionComplete *s)
- const UpnpString * TestClass_get_TheString (const TestClass *p)
- int TestClass_set_TheString (TestClass *p, const UpnpString *s)
- size_t TestClass_get_TheString_Length (const TestClass *p)
- const char * TestClass_get_TheString_cstr (const TestClass *p)
- int TestClass_strcpy_TheString (TestClass *p, const char *s)
- int TestClass_strncpy_TheString (TestClass *p, const char *s, size_t n)
- void TestClass_clear_TheString (TestClass *p)
- const DOMString TestClass_get_TheDomString (const TestClass *p)
- int TestClass_set_TheDomString (TestClass *p, const DOMString s)
- const char * TestClass_get_TheDomString_cstr (const TestClass *p)

9.20.1 Detailed Description

Source file for TestClass methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.20.2 Function Documentation

9.20.2.1 TestClass_add_to_list_TheList()

```
void TestClass_add_to_list_TheList (
            TestClass * p,
            UpnpListHead * head )
```

TestClass_add_to_list_TheList

9.20.2.2 TestClass_assign()

```
int TestClass_assign (
            TestClass * p,
            const TestClass * q )
```

Assignment operator

References TestClass_get_TheActionComplete(), TestClass_get_TheBuffer(), TestClass_get_TheCharPointer(), TestClass_get_TheDomString(), TestClass_get_TheInteger(), TestClass_get_TheList(), TestClass_get_TheString(), TestClass_get_TheUnsignedLong(), TestClass_set_TheActionComplete(), TestClass_set_TheBuffer(), TestClass_set_TheCharPointe TestClass_set_TheDomString(), TestClass_set_TheInteger(), TestClass_set_TheList(), TestClass_set_TheString(), and TestClass_set_TheUnsignedLong().

Referenced by TestClass_dup().

9.20.2.3 TestClass_clear_TheBuffer()

```
void TestClass_clear_TheBuffer (
            TestClass * p )
```

TestClass_get_TheBuffer

9.20.2.4 TestClass_clear_TheString()

```
void TestClass_clear_TheString (
            TestClass * p )
```

TestClass_clear_TheString

References UpnpString_clear().

9.20.2.5 TestClass_delete()

```
void TestClass_delete (
            TestClass * p )
```

Destructor

References ixmlFreeDOMString(), UpnpActionComplete_delete(), and UpnpString_delete().

9.20.2.6 TestClass_dup()

```
TestClass * TestClass_dup (
            const TestClass * p )
```

Copy Constructor

References TestClass_assign(), and TestClass_new().

9.20.2.7 TestClass_get_TheActionComplete()

```
const UpnpActionComplete * TestClass_get_TheActionComplete (
            const TestClass * p )
```

TestClass_get_TheActionComplete

Referenced by TestClass_assign().

9.20.2.8 TestClass_get_TheBuffer()

```
const struct TheStruct * TestClass_get_TheBuffer (
            const TestClass * p )
```

TestClass_get_TheBuffer

Referenced by TestClass_assign().

9.20.2.9 TestClass_get_TheCharPointer()

```
char * TestClass_get_TheCharPointer (
            const TestClass * p )
```

TestClass_get_TheCharPointer

Referenced by TestClass_assign().

9.20.2.10 TestClass_get_TheDomString()

```
const DOMString TestClass_get_TheDomString (
            const TestClass * p )
```

TestClass_get_TheDomString

Referenced by TestClass_assign(), and TestClass_get_TheDomString_cstr().

9.20.2.11 TestClass_get_TheDomString_cstr()

```
const char * TestClass_get_TheDomString_cstr (
            const TestClass * p )
```

TestClass_get_TheDomString_cstr

References TestClass_get_TheDomString().

9.20.2.12 TestClass_get_TheInteger()

```
int TestClass_get_TheInteger (
            const TestClass * p )
```

TestClass_get_TheInteger

Referenced by TestClass_assign().

9.20.2.13 TestClass_get_TheList()

```
const UpnpListHead * TestClass_get_TheList (
            const TestClass * p )
```

TestClass_get_TheList

Referenced by TestClass_assign().

9.20.2.14 TestClass_get_TheString()

```
const UpnpString * TestClass_get_TheString (
            const TestClass * p )
```

TestClass_get_TheString

Referenced by TestClass_assign(), TestClass_get_TheString_cstr(), and TestClass_get_TheString_Length().

9.20.2.15 TestClass_get_TheString_cstr()

```
const char * TestClass_get_TheString_cstr (
            const TestClass * p )
```

TestClass_get_TheString_cstr

References TestClass_get_TheString(), and UpnpString_get_String().

9.20.2.16 TestClass_get_TheString_Length()

```
size_t TestClass_get_TheString_Length (
            const TestClass * p )
```

TestClass_get_TheString_Length

References TestClass_get_TheString(), and UpnpString_get_Length().

9.20.2.17 TestClass_get_TheUnsignedLong()

```
unsigned long TestClass_get_TheUnsignedLong (
            const TestClass * p )
```

TestClass_get_TheUnsignedLong

Referenced by TestClass_assign().

9.20.2.18 TestClass_new()

```
TestClass * TestClass_new ( )
```

Constructor

References UpnpActionComplete_new(), and UpnpString_new().

Referenced by TestClass_dup().

9.20.2.19 TestClass_set_TheActionComplete()

```
int TestClass_set_TheActionComplete (
            TestClass * p,
            const UpnpActionComplete * n )
```

TestClass_set_TheActionComplete

References UpnpActionComplete_delete(), and UpnpActionComplete_dup().

Referenced by TestClass_assign().

9.20.2.20 TestClass_set_TheBuffer()

```
int TestClass_set_TheBuffer (
            TestClass * p,
            const struct TheStruct * buf )
```

TestClass_get_TheBuffer

Referenced by TestClass_assign().

9.20.2.21 TestClass_set_TheCharPointer()

```
int TestClass_set_TheCharPointer (
            TestClass * p,
            char * n )
```

TestClass_set_TheCharPointer

Referenced by TestClass_assign().

9.20.2.22 TestClass_set_TheDomString()

```
int TestClass_set_TheDomString (
            TestClass * p,
            const DOMString s )
```

TestClass_set_TheDomString

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by TestClass_assign().

9.20.2.23 TestClass_set_TheInteger()

```
int TestClass_set_TheInteger (
            TestClass * p,
            int n )
```

TestClass_set_TheInteger

Referenced by TestClass_assign().

9.20.2.24 TestClass_set_TheList()

```
int TestClass_set_TheList (
            TestClass * p,
            const UpnpListHead * q )
```

TestClass_set_TheList

Referenced by TestClass_assign().

9.20.2.25 TestClass_set_TheString()

```
int TestClass_set_TheString (
            TestClass * p,
            const UpnpString * s )
```

TestClass_set_TheString

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by TestClass_assign().

9.20.2.26 TestClass_set_TheUnsignedLong()

```
int TestClass_set_TheUnsignedLong (
            TestClass * p,
            unsigned long n )
```

TestClass_set_TheUnsignedLong

Referenced by TestClass_assign().

9.20.2.27 TestClass_strcpy_TheString()

```
int TestClass_strcpy_TheString (
            TestClass * p,
            const char * s )
```

TestClass_strcpy_TheString

References UpnpString_set_String().

9.20.2.28 TestClass_strncpy_TheString()

```
int TestClass_strncpy_TheString (
            TestClass * p,
            const char * s,
            size_t n )
```

TestClass_strncpy_TheString

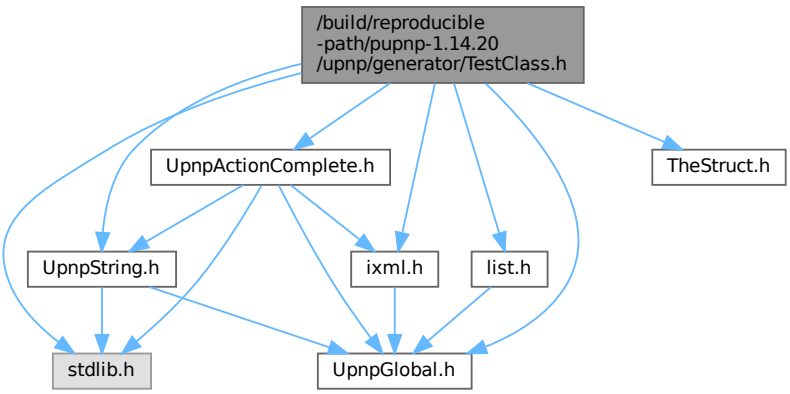
References UpnpString_set_StringN().

9.21 /build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.h File Reference

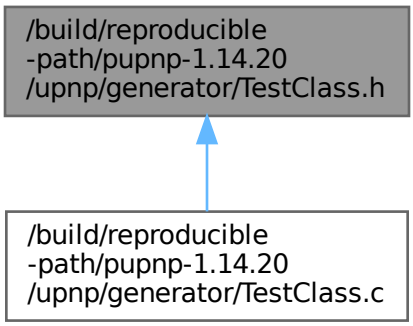
Header file for TestClass methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "TheStruct.h"
#include "UpnpActionComplete.h"
#include "UpnpString.h"
#include "ixml.h"
#include "list.h"
```

Include dependency graph for TestClass.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_TestClass TestClass

Functions

- UPNP_EXPORT_SPEC TestClass * TestClass_new ()
- UPNP_EXPORT_SPEC void TestClass_delete (TestClass *p)
- UPNP_EXPORT_SPEC TestClass * TestClass_dup (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_assign (TestClass *p, const TestClass *q)
- UPNP_EXPORT_SPEC const UpnpListHead * TestClass_get_TheList (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheList (TestClass *p, const UpnpListHead *q)
- UPNP_EXPORT_SPEC void TestClass_add_to_list_TheList (TestClass *p, UpnpListHead *head)
- UPNP_EXPORT_SPEC int TestClass_get_TheInteger (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheInteger (TestClass *p, int n)
- UPNP_EXPORT_SPEC unsigned long TestClass_get_TheUnsignedLong (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheUnsignedLong (TestClass *p, unsigned long n)
- UPNP_EXPORT_SPEC char * TestClass_get_TheCharPointer (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheCharPointer (TestClass *p, char *n)
- UPNP_EXPORT_SPEC const struct TheStruct * TestClass_get_TheBuffer (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheBuffer (TestClass *p, const struct TheStruct *buf)
- UPNP_EXPORT_SPEC void TestClass_clear_TheBuffer (TestClass *p)
- UPNP_EXPORT_SPEC const UpnpActionComplete * TestClass_get_TheActionComplete (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheActionComplete (TestClass *p, const UpnpActionComplete *n)
- UPNP_EXPORT_SPEC const UpnpString * TestClass_get_TheString (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheString (TestClass *p, const UpnpString *s)
- UPNP_EXPORT_SPEC size_t TestClass_get_TheString_Length (const TestClass *p)
- UPNP_EXPORT_SPEC const char * TestClass_get_TheString_cstr (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_strcpy_TheString (TestClass *p, const char *s)
- UPNP_EXPORT_SPEC int TestClass_strncpy_TheString (TestClass *p, const char *s, size_t n)
- UPNP_EXPORT_SPEC void TestClass_clear_TheString (TestClass *p)
- UPNP_EXPORT_SPEC const DOMString TestClass_get_TheDomString (const TestClass *p)
- UPNP_EXPORT_SPEC int TestClass_set_TheDomString (TestClass *p, const DOMString s)
- UPNP_EXPORT_SPEC const char * TestClass_get_TheDomString_cstr (const TestClass *p)

9.21.1 Detailed Description

Header file for TestClass methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.21.2 Typedef Documentation

9.21.2.1 TestClass

```
typedef struct s_TestClass TestClass
```

TestClass

9.21.3 Function Documentation

9.21.3.1 TestClass_add_to_list_TheList()

```
UPNP_EXPORT_SPEC void TestClass_add_to_list_TheList (
            TestClass * p,
            UpnpListHead * head )
```

TestClass_add_to_list_TheList

9.21.3.2 TestClass_assign()

```
UPNP_EXPORT_SPEC int TestClass_assign (
            TestClass * p,
            const TestClass * q )
```

Assignment operator

References TestClass_get_TheActionComplete(), TestClass_get_TheBuffer(), TestClass_get_TheCharPointer(), TestClass_get_TheDomString(), TestClass_get_TheInteger(), TestClass_get_TheList(), TestClass_get_TheString(), TestClass_get_TheUnsignedLong(), TestClass_set_TheActionComplete(), TestClass_set_TheBuffer(), TestClass_set_TheCharPointe TestClass_set_TheDomString(), TestClass_set_TheInteger(), TestClass_set_TheList(), TestClass_set_TheString(), and TestClass_set_TheUnsignedLong().

Referenced by TestClass_dup().

9.21.3.3 TestClass_clear_TheBuffer()

```
UPNP_EXPORT_SPEC void TestClass_clear_TheBuffer (
            TestClass * p )
```

TestClass_get_TheBuffer

9.21.3.4 TestClass_clear_TheString()

```
UPNP_EXPORT_SPEC void TestClass_clear_TheString (
            TestClass * p )
```

TestClass_clear_TheString

References UpnpString_clear().

9.21.3.5 TestClass_delete()

```
UPNP_EXPORT_SPEC void TestClass_delete (
            TestClass * p )
```

Destructor

References ixmlFreeDOMString(), UpnpActionComplete_delete(), and UpnpString_delete().

9.21.3.6 TestClass_dup()

```
UPNP_EXPORT_SPEC TestClass * TestClass_dup (
            const TestClass * p )
```

Copy Constructor

References TestClass_assign(), and TestClass_new().

9.21.3.7 TestClass_get_TheActionComplete()

```
UPNP_EXPORT_SPEC const UpnpActionComplete * TestClass_get_TheActionComplete (
            const TestClass * p )
```

TestClass_get_TheActionComplete

Referenced by TestClass_assign().

9.21.3.8 TestClass_get_TheBuffer()

```
UPNP_EXPORT_SPEC const struct TheStruct * TestClass_get_TheBuffer (
            const TestClass * p )
```

TestClass_get_TheBuffer

Referenced by TestClass_assign().

9.21.3.9 TestClass_get_TheCharPointer()

```
UPNP_EXPORT_SPEC char * TestClass_get_TheCharPointer (
            const TestClass * p )
```

TestClass_get_TheCharPointer

Referenced by TestClass_assign().

9.21.3.10 TestClass_get_TheDomString()

```
UPNP_EXPORT_SPEC const DOMString TestClass_get_TheDomString (
            const TestClass * p )
```

TestClass_get_TheDomString

Referenced by TestClass_assign(), and TestClass_get_TheDomString_cstr().

9.21.3.11 TestClass_get_TheDomString_cstr()

```
UPNP_EXPORT_SPEC const char * TestClass_get_TheDomString_cstr (
            const TestClass * p )
```

TestClass_get_TheDomString_cstr

References TestClass_get_TheDomString().

9.21.3.12 TestClass_get_TheInteger()

```
UPNP_EXPORT_SPEC int TestClass_get_TheInteger (
            const TestClass * p )
```

TestClass_get_TheInteger

Referenced by TestClass_assign().

9.21.3.13 TestClass_get_TheList()

```
UPNP_EXPORT_SPEC const UpnpListHead * TestClass_get_TheList (
            const TestClass * p )
```

TestClass_get_TheList

Referenced by TestClass_assign().

9.21.3.14 TestClass_get_TheString()

```
UPNP_EXPORT_SPEC const UpnpString * TestClass_get_TheString (
            const TestClass * p )
```

TestClass_get_TheString

Referenced by TestClass_assign(), TestClass_get_TheString_cstr(), and TestClass_get_TheString_Length().

9.21.3.15 TestClass_get_TheString_cstr()

```
UPNP_EXPORT_SPEC const char * TestClass_get_TheString_cstr (
            const TestClass * p )
```

TestClass_get_TheString_cstr

References TestClass_get_TheString(), and UpnpString_get_String().

9.21.3.16 TestClass_get_TheString_Length()

```
UPNP_EXPORT_SPEC size_t TestClass_get_TheString_Length (
            const TestClass * p )
```

TestClass_get_TheString_Length

References TestClass_get_TheString(), and UpnpString_get_Length().

9.21.3.17 TestClass_get_TheUnsignedLong()

```
UPNP_EXPORT_SPEC unsigned long TestClass_get_TheUnsignedLong (
            const TestClass * p )
```

TestClass_get_TheUnsignedLong

Referenced by TestClass_assign().

9.21.3.18 TestClass_new()

```
UPNP_EXPORT_SPEC TestClass * TestClass_new ( )
```

Constructor

References UpnpActionComplete_new(), and UpnpString_new().

Referenced by TestClass_dup().

9.21.3.19 TestClass_set_TheActionComplete()

```
UPNP_EXPORT_SPEC int TestClass_set_TheActionComplete (
            TestClass * p,
            const UpnpActionComplete * n )
```

TestClass_set_TheActionComplete

References UpnpActionComplete_delete(), and UpnpActionComplete_dup().

Referenced by TestClass_assign().

9.21.3.20 TestClass_set_TheBuffer()

```
UPNP_EXPORT_SPEC int TestClass_set_TheBuffer (
            TestClass * p,
            const struct TheStruct * buf )
```

TestClass_get_TheBuffer

Referenced by TestClass_assign().

9.21.3.21 TestClass_set_TheCharPointer()

```
UPNP_EXPORT_SPEC int TestClass_set_TheCharPointer (
            TestClass * p,
            char * n )
```

TestClass_set_TheCharPointer

Referenced by TestClass_assign().

9.21.3.22 TestClass_set_TheDomString()

```
UPNP_EXPORT_SPEC int TestClass_set_TheDomString (
            TestClass * p,
            const DOMString s )
```

TestClass_set_TheDomString

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by TestClass_assign().

9.21.3.23 TestClass_set_TheInteger()

```
UPNP_EXPORT_SPEC int TestClass_set_TheInteger (
            TestClass * p,
            int n )
```

TestClass_set_TheInteger

Referenced by TestClass_assign().

9.21.3.24 TestClass_set_TheList()

```
UPNP_EXPORT_SPEC int TestClass_set_TheList (
            TestClass * p,
            const UpnpListHead * q )
```

TestClass_set_TheList

Referenced by TestClass_assign().

9.21.3.25 TestClass_set_TheString()

```
UPNP_EXPORT_SPEC int TestClass_set_TheString (
            TestClass * p,
            const UpnpString * s )
```

TestClass_set_TheString

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by TestClass_assign().

9.21.3.26 TestClass_set_TheUnsignedLong()

```
UPNP_EXPORT_SPEC int TestClass_set_TheUnsignedLong (
            TestClass * p,
            unsigned long n )
```

TestClass_set_TheUnsignedLong

Referenced by TestClass_assign().

9.21.3.27 TestClass_strcpy_TheString()

```
UPNP_EXPORT_SPEC int TestClass_strcpy_TheString (
            TestClass * p,
            const char * s )
```

TestClass_strcpy_TheString

References UpnpString_set_String().

9.21.3.28 TestClass_strncpy_TheString()

```
UPNP_EXPORT_SPEC int TestClass_strncpy_TheString (
            TestClass * p,
            const char * s,
            size_t n )
```

TestClass_strncpy_TheString

References UpnpString_set_StringN().

9.22 TestClass.h

Go to the documentation of this file.

```
00001 #ifndef TESTCLASS_H
00002 #define TESTCLASS_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "TheStruct.h"
00019 #include "UpnpActionComplete.h"
00020 #include "UpnpString.h"
00021 #include "ixml.h"
00022 #include "list.h"
00023
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif /* __cplusplus */
00027
00031 typedef struct s_TestClass TestClass;
00032
00034 UPNP_EXPORT_SPEC TestClass *TestClass_new();
00036 UPNP_EXPORT_SPEC void TestClass_delete(TestClass *p);
00038 UPNP_EXPORT_SPEC TestClass *TestClass_dup(const TestClass *p);
00040 UPNP_EXPORT_SPEC int TestClass_assign(TestClass *p, const TestClass *q);
00041
00043 UPNP_EXPORT_SPEC const UpnpListHead *TestClass_get_TheList(const TestClass *p);
00045 UPNP_EXPORT_SPEC int TestClass_set_TheList(TestClass *p, const UpnpListHead *q);
00047 UPNP_EXPORT_SPEC void TestClass_add_to_list_TheList(
```

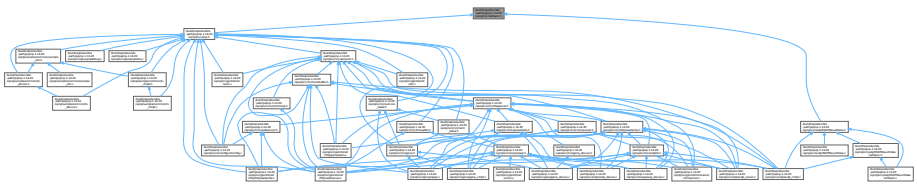
```
00048         TestClass *p, UpnpListHead *head);
00049
00051 UPNP_EXPORT_SPEC int TestClass_get_TheInteger(const TestClass *p);
00053 UPNP_EXPORT_SPEC int TestClass_set_TheInteger(TestClass *p, int n);
00054
00056 UPNP_EXPORT_SPEC unsigned long TestClass_get_TheUnsignedLong(
00057         const TestClass *p);
00059 UPNP_EXPORT_SPEC int TestClass_set_TheUnsignedLong(
00060         TestClass *p, unsigned long n);
00061
00063 UPNP_EXPORT_SPEC char *TestClass_get_TheCharPointer(const TestClass *p);
00065 UPNP_EXPORT_SPEC int TestClass_set_TheCharPointer(TestClass *p, char *n);
00066
00068 UPNP_EXPORT_SPEC const struct TheStruct *TestClass_get_TheBuffer(
00069         const TestClass *p);
00071 UPNP_EXPORT_SPEC int TestClass_set_TheBuffer(
00072         TestClass *p, const struct TheStruct *buf);
00074 UPNP_EXPORT_SPEC void TestClass_clear_TheBuffer(TestClass *p);
00075
00077 UPNP_EXPORT_SPEC const UpnpActionComplete *TestClass_get_TheActionComplete(
00078         const TestClass *p);
00080 UPNP_EXPORT_SPEC int TestClass_set_TheActionComplete(
00081         TestClass *p, const UpnpActionComplete *n);
00082
00084 UPNP_EXPORT_SPEC const UpnpString *TestClass_get_TheString(const TestClass *p);
00086 UPNP_EXPORT_SPEC int TestClass_set_TheString(TestClass *p, const UpnpString *s);
00088 UPNP_EXPORT_SPEC size_t TestClass_get_TheString_Length(const TestClass *p);
00090 UPNP_EXPORT_SPEC const char *TestClass_get_TheString_cstr(const TestClass *p);
00092 UPNP_EXPORT_SPEC int TestClass_strcpy_TheString(TestClass *p, const char *s);
00094 UPNP_EXPORT_SPEC int TestClass_strncpy_TheString(
00095         TestClass *p, const char *s, size_t n);
00097 UPNP_EXPORT_SPEC void TestClass_clear_TheString(TestClass *p);
00098
00100 UPNP_EXPORT_SPEC const DOMString TestClass_get_TheDomString(const TestClass *p);
00102 UPNP_EXPORT_SPEC int TestClass_set_TheDomString(
00103         TestClass *p, const DOMString s);
00105 UPNP_EXPORT_SPEC const char *TestClass_get_TheDomString_cstr(
00106         const TestClass *p);
00107
00108 #ifdef __cplusplus
00109 }
00110 #endif /* __cplusplus */
00111
00112 #endif /* TESTCLASS_H */
```

9.23 TheStruct.h

```
00001 struct TheStruct
00002 {
00003         int x;
00004         int y;
00005 };
```

9.24 /build/reproducible-path/pupnp-1.14.20/upnp/inc/Callback.h File Reference

This graph shows which files directly or indirectly include this file:



Typedefs

- typedef enum Upnp_EventType_e **Upnp_EventType**
- typedef int(∗ Upnp_FunPtr) (Upnp_EventType EventType, const void ∗Event, void ∗Cookie)

Enumerations

- enum Upnp_EventType_e {
  UPNP_CONTROL_ACTION_REQUEST , UPNP_CONTROL_ACTION_COMPLETE , UPNP_CONTROL_GET_VAR_REQUES
  , UPNP_CONTROL_GET_VAR_COMPLETE ,
  UPNP_DISCOVERY_ADVERTISEMENT_ALIVE , UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE ,
  UPNP_DISCOVERY_SEARCH_RESULT , UPNP_DISCOVERY_SEARCH_TIMEOUT ,
  UPNP_EVENT_SUBSCRIPTION_REQUEST , UPNP_EVENT_RECEIVED , UPNP_EVENT_RENEWAL_COMPLETE
  , UPNP_EVENT_SUBSCRIBE_COMPLETE ,
  UPNP_EVENT_UNSUBSCRIBE_COMPLETE , UPNP_EVENT_AUTORENEWAL_FAILED , UPNP_EVENT_SUBSCRIPTION_
  }

  The reason code for an event callback.

9.24.1 Typedef Documentation

9.24.1.1 Upnp_FunPtr

```
typedef int(* Upnp_FunPtr) ( Upnp_EventType EventType, const void *Event, void *Cookie)
```

All callback functions share the same prototype, documented below. Note that any memory passed to the callback function is valid only during the callback and should be copied if it needs to persist. This callback function needs to be thread safe. The context of the callback is always on a valid thread context and standard synchronization methods can be used. Note, however, because of this the callback cannot call SDK functions unless explicitly noted.

```
int CallbackFxn(Upnp_EventType EventType, void *Event, void *Cookie);
```

where **EventType** is the event that triggered the callback, **Event** is a structure that denotes event-specific information for that event, and **Cookie** is the user data passed when the callback was registered.

See **Upnp_EventType** for more information on the callback values and the associated **Event** parameter.

The return value of the callback is currently ignored. It may be used in the future to communicate results back to the SDK.

Parameters

in	*EventType*	.
in	*Event*	.
in	*Cookie*	.

9.24.2 Enumeration Type Documentation

9.24.2.1 Upnp_EventType_e

```
enum Upnp_EventType_e
```

The reason code for an event callback.

The **Event** parameter will be different depending on the reason for the callback. The descriptions for each event type describe the contents of the **Event** parameter.

Enumerator

UPNP_CONTROL_ACTION_REQUEST	Received by a device when a control point issues a control request. The **Event** parameter contains a pointer to a **UpnpActionRequest** structure containing the action. The application stores the results of the action in this structure.
UPNP_CONTROL_ACTION_COMPLETE	A **UpnpSendActionAsync** call completed. The **Event** parameter contains a pointer to a **UpnpActionComplete** structure with the results of the action.
UPNP_CONTROL_GET_VAR_REQUEST	Received by a device when a query for a single service variable arrives. The **Event** parameter contains a pointer to a **UpnpStateVarRequest** structure containing the name of the variable and value.
UPNP_CONTROL_GET_VAR_COMPLETE	A **UpnpGetServiceVarStatus** call completed. The **Event** parameter contains a pointer to a **UpnpStateVarComplete** structure containing the value for the variable.
UPNP_DISCOVERY_ADVERTISEMENT_ALIVE	Received by a control point when a new device or service is available. The **Event** parameter contains a pointer to a **UpnpDiscovery** structure with the information about the device or service.
UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE	Received by a control point when a device or service shuts down. The **Event** parameter contains a pointer to a **UpnpDiscovery** structure containing the information about the device or service.
UPNP_DISCOVERY_SEARCH_RESULT	Received by a control point when a matching device or service responds. The **Event** parameter contains a pointer to a **UpnpDiscovery** structure containing the information about the reply to the search request.
UPNP_DISCOVERY_SEARCH_TIMEOUT	Received by a control point when the search timeout expires. The SDK generates no more callbacks for this search after this event. The **Event** parameter is `NULL`.
UPNP_EVENT_SUBSCRIPTION_REQUEST	Received by a device when a subscription arrives. The **Event** parameter contains a pointer to a **UpnpSubscriptionRequest** structure. At this point, the subscription has already been accepted. **UpnpAcceptSubscription** needs to be called to confirm the subscription and transmit the initial state table. This can be done during this callback. The SDK generates no events for a subscription unless the device application calls **UpnpAcceptSubscription**.
UPNP_EVENT_RECEIVED	Received by a control point when an event arrives. The **Event** parameter contains a **UpnpEvent** structure with the information about the event.

Enumerator

UPNP_EVENT_RENEWAL_COMPLETE	A **UpnpRenewSubscriptionAsync** call completed. The status of the renewal is in the **Event** parameter as a **Upnp_Event_Subscription** structure.
UPNP_EVENT_SUBSCRIBE_COMPLETE	A **UpnpSubscribeAsync** call completed. The status of the subscription is in the **Event** parameter as a **Upnp_Event_Subscription** structure.
UPNP_EVENT_UNSUBSCRIBE_COMPLETE	A **UpnpUnSubscribeAsync** call completed. The status of the subscription is in the **Event** parameter as a **UpnpEventSubscribe** structure.
UPNP_EVENT_AUTORENEWAL_FAILED	The auto-renewal of a client subscription failed. The **Event** parameter is a **UpnpEventSubscribe** structure with the error code set appropriately. The subscription is no longer valid.
UPNP_EVENT_SUBSCRIPTION_EXPIRED	A client subscription has expired. This will only occur if auto-renewal of subscriptions is disabled. The **Event** parameter is a **UpnpEventSubscribe** structure. The subscription is no longer valid.

9.25 Callback.h

Go to the documentation of this file.

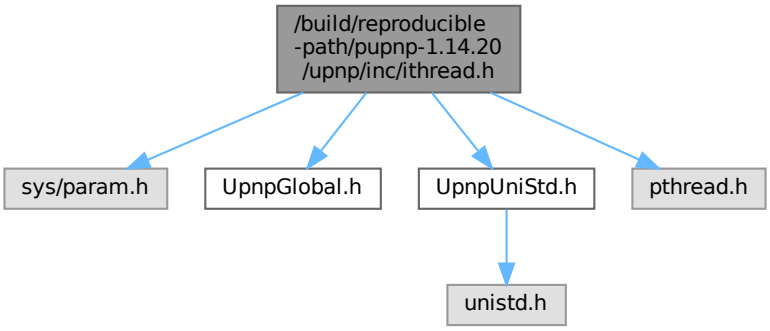
```
00001
00002 #ifndef CALLBACK_H
00003 #define CALLBACK_H
00004
00016 enum Upnp_EventType_e
00017 {
00018         /*
00019          * Control callbacks
00020          */
00021
00026         UPNP_CONTROL_ACTION_REQUEST,
00027
00031         UPNP_CONTROL_ACTION_COMPLETE,
00032
00037         UPNP_CONTROL_GET_VAR_REQUEST,
00038
00042         UPNP_CONTROL_GET_VAR_COMPLETE,
00043
00044         /*
00045          * Discovery callbacks
00046          */
00047
00052         UPNP_DISCOVERY_ADVERTISEMENT_ALIVE,
00053
00057         UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE,
00058
00063         UPNP_DISCOVERY_SEARCH_RESULT,
00064
00068         UPNP_DISCOVERY_SEARCH_TIMEOUT,
00069
00070         /*
00071          * Eventing callbacks
00072          */
00073
00083         UPNP_EVENT_SUBSCRIPTION_REQUEST,
00084
00088         UPNP_EVENT_RECEIVED,
00089
00093         UPNP_EVENT_RENEWAL_COMPLETE,
00094
00098         UPNP_EVENT_SUBSCRIBE_COMPLETE,
00099
```

```
00103         UPNP_EVENT_UNSUBSCRIBE_COMPLETE,
00104
00109         UPNP_EVENT_AUTORENEWAL_FAILED,
00110
00115         UPNP_EVENT_SUBSCRIPTION_EXPIRED
00116 };
00117
00118 typedef enum Upnp_EventType_e Upnp_EventType;
00119
00145 typedef int (*Upnp_FunPtr)(
00147         Upnp_EventType EventType,
00149         const void *Event,
00151         void *Cookie);
00152
00153 #endif /* CALLBACK_H */
```

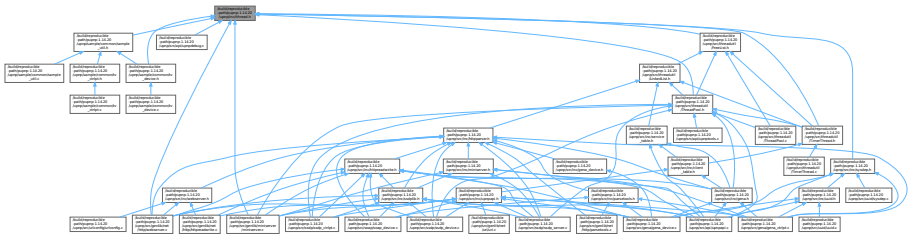
9.26 /build/reproducible-path/pupnp-1.14.20/upnp/inc/ithread.h File Reference

```
#include <sys/param.h>
#include "UpnpGlobal.h"
#include "UpnpUniStd.h"
#include <pthread.h>
```

Include dependency graph for ithread.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **ITHREAD_MUTEX_FAST_NP** PTHREAD_MUTEX_FAST_NP

- #define **ITHREAD_MUTEX_RECURSIVE_NP** PTHREAD_MUTEX_RECURSIVE_NP
- #define **ITHREAD_MUTEX_ERRORCHECK_NP** PTHREAD_MUTEX_ERRORCHECK_NP
- #define **ITHREAD_PROCESS_PRIVATE** PTHREAD_PROCESS_PRIVATE
- #define **ITHREAD_PROCESS_SHARED** PTHREAD_PROCESS_SHARED
- #define **ITHREAD_CANCELED** PTHREAD_CANCELED
- #define **ITHREAD_STACK_MIN** PTHREAD_STACK_MIN
- #define **ITHREAD_CREATE_DETACHED** PTHREAD_CREATE_DETACHED
- #define **ITHREAD_CREATE_JOINABLE** PTHREAD_CREATE_JOINABLE
- #define **ithread_mutexattr_init** pthread_mutexattr_init
- #define **ithread_mutexattr_destroy** pthread_mutexattr_destroy
- #define **ithread_mutexattr_setkind_np** pthread_mutexattr_settype
- #define **ithread_mutexattr_settype** pthread_mutexattr_settype
- #define **ithread_mutexattr_getkind_np** pthread_mutexattr_gettype
- #define **ithread_mutexattr_gettype** pthread_mutexattr_gettype
- #define **ithread_mutex_init** pthread_mutex_init
- #define **ithread_mutex_lock** pthread_mutex_lock
- #define **ithread_mutex_unlock** pthread_mutex_unlock
- #define **ithread_mutex_destroy** pthread_mutex_destroy
- #define **ithread_rwlock_init** ithread_mutex_init
- #define **ithread_rwlock_rdlock** ithread_mutex_lock
- #define **ithread_rwlock_wrlock** ithread_mutex_lock
- #define **ithread_rwlock_unlock** ithread_mutex_unlock
- #define **ithread_rwlock_destroy** ithread_mutex_destroy
- #define **ithread_cond_init** pthread_cond_init
- #define **ithread_cond_signal** pthread_cond_signal
- #define **ithread_cond_broadcast** pthread_cond_broadcast
- #define **ithread_cond_wait** pthread_cond_wait
- #define **ithread_cond_timedwait** pthread_cond_timedwait
- #define **ithread_cond_destroy** pthread_cond_destroy
- #define **ithread_attr_init** pthread_attr_init
- #define **ithread_attr_destroy** pthread_attr_destroy
- #define **ithread_attr_setstacksize** pthread_attr_setstacksize
- #define **ithread_attr_setdetachstate** pthread_attr_setdetachstate
- #define **ithread_create** pthread_create
- #define **ithread_cancel** pthread_cancel
- #define **ithread_exit** pthread_exit
- #define **ithread_get_current_thread_id** pthread_self
- #define **ithread_self** pthread_self
- #define **ithread_detach** pthread_detach
- #define **ithread_join** pthread_join
- #define **isleep** sleep
- #define **imillisleep**(x) usleep(1000 ∗ x)

Typedefs

- typedef pthread_t **ithread_t**
- typedef pthread_attr_t **ithread_attr_t**
- typedef void(∗ **start_routine**) (void ∗arg)
- typedef pthread_cond_t **ithread_cond_t**
- typedef pthread_mutexattr_t **ithread_mutexattr_t**
- typedef pthread_mutex_t **ithread_mutex_t**
- typedef pthread_condattr_t **ithread_condattr_t**
- typedef ithread_mutex_t **ithread_rwlock_t**

Functions

- static int **ithread_initialize_library** (void)
- static int **ithread_cleanup_library** (void)
- static int **ithread_initialize_thread** (void)
- static int **ithread_cleanup_thread** (void)
- int **pthread_mutexattr_setkind_np** (pthread_mutexattr_t *attr, int kind)

9.27 ithread.h

Go to the documentation of this file.

```
00001 #ifndef ITHREAD_H
00002 #define ITHREAD_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  * Copyright (c) 2012 France Telecom All rights reserved.
00009  *
00010  * Redistribution and use in source and binary forms, with or without
00011  * modification, are permitted provided that the following conditions are met:
00012  *
00013  * * Redistributions of source code must retain the above copyright notice,
00014  * this list of conditions and the following disclaimer.
00015  * * Redistributions in binary form must reproduce the above copyright notice,
00016  * this list of conditions and the following disclaimer in the documentation
00017  * and/or other materials provided with the distribution.
00018  * * Neither name of Intel Corporation nor the names of its contributors
00019  * may be used to endorse or promote products derived from this software
00020  * without specific prior written permission.
00021  *
00022  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00026  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00027  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00028  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00029  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00030  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00031  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00032  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  ******************************************************************************/
00035
00040 #if !defined(_WIN32)
00041         #include <sys/param.h>
00042 #endif
00043
00044 #include "UpnpGlobal.h" /* For UPNP_INLINE, UPNP_EXPORT_SPEC */
00045 #include "UpnpUniStd.h" /* for close() */
00046
00047 #ifdef __cplusplus
00048 extern "C" {
00049 #endif
00050
00051 #include <pthread.h>
00052
00053 #if defined(BSD) && !defined(__GNU__)
00054         #define PTHREAD_MUTEX_RECURSIVE_NP PTHREAD_MUTEX_RECURSIVE
00055 #endif
00056
00057 #if defined(PTHREAD_MUTEX_RECURSIVE) || defined(__DragonFly__)
00058         /* This system has SuS2-compliant mutex attributes.
00059          * E.g. on Cygwin, where we don't have the old nonportable (NP) symbols
00060          */
00061         #define ITHREAD_MUTEX_FAST_NP PTHREAD_MUTEX_NORMAL
00062         #define ITHREAD_MUTEX_RECURSIVE_NP PTHREAD_MUTEX_RECURSIVE
00063         #define ITHREAD_MUTEX_ERRORCHECK_NP PTHREAD_MUTEX_ERRORCHECK
00064 #else /* PTHREAD_MUTEX_RECURSIVE */
00065         #define ITHREAD_MUTEX_FAST_NP PTHREAD_MUTEX_FAST_NP
00066         #define ITHREAD_MUTEX_RECURSIVE_NP PTHREAD_MUTEX_RECURSIVE_NP
00067         #define ITHREAD_MUTEX_ERRORCHECK_NP PTHREAD_MUTEX_ERRORCHECK_NP
00068 #endif /* PTHREAD_MUTEX_RECURSIVE */
00069
00070 #define ITHREAD_PROCESS_PRIVATE PTHREAD_PROCESS_PRIVATE
00071 #define ITHREAD_PROCESS_SHARED PTHREAD_PROCESS_SHARED
```

```
00072
00073 #define ITHREAD_CANCELED PTHREAD_CANCELED
00074
00075 #define ITHREAD_STACK_MIN PTHREAD_STACK_MIN
00076 #define ITHREAD_CREATE_DETACHED PTHREAD_CREATE_DETACHED
00077 #define ITHREAD_CREATE_JOINABLE PTHREAD_CREATE_JOINABLE
00078
00079 /*****************************************************************************
00080  * Name: ithread_t
00081  *
00082  *  Description:
00083  *      Thread handle.
00084  *      typedef to pthread_t.
00085  *      Internal Use Only.
00086  ****************************************************************************/
00087 typedef pthread_t ithread_t;
00088
00089 /*****************************************************************************
00090  * Name: ithread_attr_t
00091  *
00092  *  Description:
00093  *      Thread attribute.
00094  *      typedef to pthread_attr_t
00095  *      Internal Use Only
00096  ****************************************************************************/
00097 typedef pthread_attr_t ithread_attr_t;
00098
00099 /*****************************************************************************
00100  * Name: start_routine
00101  *
00102  *  Description:
00103  *      Thread start routine
00104  *      Internal Use Only.
00105  ****************************************************************************/
00106 typedef void (*start_routine)(void *arg);
00107
00108 /*****************************************************************************
00109  * Name: ithread_cond_t
00110  *
00111  *  Description:
00112  *      condition variable.
00113  *      typedef to pthread_cond_t
00114  *      Internal Use Only.
00115  ****************************************************************************/
00116 typedef pthread_cond_t ithread_cond_t;
00117
00118 /*****************************************************************************
00119  * Name: ithread_mutexattr_t
00120  *
00121  *  Description:
00122  *      Mutex attribute.
00123  *      typedef to pthread_mutexattr_t
00124  *      Internal Use Only
00125  ****************************************************************************/
00126 typedef pthread_mutexattr_t ithread_mutexattr_t;
00127
00128 /*****************************************************************************
00129  * Name: ithread_mutex_t
00130  *
00131  *  Description:
00132  *      Mutex.
00133  *      typedef to pthread_mutex_t
00134  *      Internal Use Only.
00135  ****************************************************************************/
00136 typedef pthread_mutex_t ithread_mutex_t;
00137
00138 /*****************************************************************************
00139  * Name: ithread_condattr_t
00140  *
00141  *  Description:
00142  *      Condition attribute.
00143  *      typedef to pthread_condattr_t
00144  *      NOT USED
00145  *      Internal Use Only
00146  ****************************************************************************/
00147 typedef pthread_condattr_t ithread_condattr_t;
00148
00149 /*****************************************************************************
00150  * Name: ithread_rwlockattr_t
00151  *
00152  *  Description:
00153  *      Mutex attribute.
00154  *      typedef to pthread_rwlockattr_t
00155  *      Internal Use Only
00156  ****************************************************************************/
00157 #if UPNP_USE_RWLOCK
00158 typedef pthread_rwlockattr_t ithread_rwlockattr_t;
```

```
00159 #endif /* UPNP_USE_RWLOCK */
00160
00161 /************************************************************************
00162  * Name: ithread_rwlock_t
00163  *
00164  *  Description:
00165  *      Condition attribute.
00166  *      typedef to pthread_rwlock_t
00167  *      Internal Use Only
00168  ************************************************************************/
00169 #if UPNP_USE_RWLOCK
00170 typedef pthread_rwlock_t ithread_rwlock_t;
00171 #else
00172 /* Read-write locks aren't available: use mutex instead. */
00173 typedef ithread_mutex_t ithread_rwlock_t;
00174 #endif /* UPNP_USE_RWLOCK */
00175
00176 /************************************************************************
00177  * Function: ithread_initialize_library
00178  *
00179  *  Description:
00180  *      Initializes the library. Does nothing in all implementations, except
00181  *      when statically linked for WIN32.
00182  *  Parameters:
00183  *      none.
00184  *  Returns:
00185  *      0 on success, Nonzero on failure.
00186  ************************************************************************/
00187 static UPNP_INLINE int ithread_initialize_library(void)
00188 {
00189         int ret = 0;
00190
00191         return ret;
00192 }
00193
00194 /************************************************************************
00195  * Function: ithread_cleanup_library
00196  *
00197  *  Description:
00198  *      Clean up library resources. Does nothing in all implementations, except
00199  *      when statically linked for WIN32.
00200  *  Parameters:
00201  *      none.
00202  *  Returns:
00203  *      0 on success, Nonzero on failure.
00204  ************************************************************************/
00205 static UPNP_INLINE int ithread_cleanup_library(void)
00206 {
00207         int ret = 0;
00208
00209         return ret;
00210 }
00211
00212 /************************************************************************
00213  * Function: ithread_initialize_thread
00214  *
00215  *  Description:
00216  *      Initializes the thread. Does nothing in all implementations, except
00217  *      when statically linked for WIN32.
00218  *  Parameters:
00219  *      none.
00220  *  Returns:
00221  *      0 on success, Nonzero on failure.
00222  ************************************************************************/
00223 static UPNP_INLINE int ithread_initialize_thread(void)
00224 {
00225         int ret = 0;
00226
00227 #if defined(_WIN32) && defined(PTW32_STATIC_LIB)
00228         ret = !pthread_win32_thread_attach_np();
00229 #endif
00230
00231         return ret;
00232 }
00233
00234 /************************************************************************
00235  * Function: ithread_cleanup_thread
00236  *
00237  *  Description:
00238  *      Clean up thread resources. Does nothing in all implementations, except
00239  *      when statically linked for WIN32.
00240  *  Parameters:
00241  *      none.
00242  *  Returns:
00243  *      0 on success, Nonzero on failure.
00244  ************************************************************************/
00245 static UPNP_INLINE int ithread_cleanup_thread(void)
```

```
00246 {
00247         int ret = 0;
00248
00249 #if defined(_WIN32) && defined(PTW32_STATIC_LIB)
00250         ret = !pthread_win32_thread_detach_np();
00251 #endif
00252
00253         return ret;
00254 }
00255
00256 /*****************************************************************************
00257  * Function: ithread_mutexattr_init
00258  *
00259  *  Description:
00260  *      Initializes a mutex attribute variable.
00261  *      Used to set the type of the mutex.
00262  *  Parameters:
00263  *      ithread_mutexattr_init * attr (must be valid non NULL pointer to
00264  *                                     pthread_mutexattr_t)
00265  *  Returns:
00266  *      0 on success, Nonzero on failure.
00267  *      Always returns 0.
00268  *      See man page for pthread_mutexattr_init
00269  ****************************************************************************/
00270 #define ithread_mutexattr_init pthread_mutexattr_init
00271
00272 /*****************************************************************************
00273  * Function: ithread_mutexattr_destroy
00274  *
00275  *  Description:
00276  *      Releases any resources held by the mutex attribute.
00277  *      Currently there are no resources associated with the attribute
00278  *  Parameters:
00279  *      ithread_mutexattr_t * attr (must be valid non NULL pointer to
00280  *                                  pthread_mutexattr_t)
00281  *  Returns:
00282  *      0 on success, Nonzero on failure.
00283  *      Always returns 0.
00284  *      See man page for pthread_mutexattr_destroy
00285  ****************************************************************************/
00286 #define ithread_mutexattr_destroy pthread_mutexattr_destroy
00287
00288 /*****************************************************************************
00289  * Function: ithread_mutexattr_setkind_np
00290  *
00291  *  Description:
00292  *      Sets the mutex type in the attribute.
00293  *      Valid types are: ITHREAD_MUTEX_FAST_NP
00294  *                       ITHREAD_MUTEX_RECURSIVE_NP
00295  *                       ITHREAD_MUTEX_ERRORCHECK_NP
00296  *
00297  *  Parameters:
00298  *      ithread_mutexattr_t * attr (must be valid non NULL pointer to
00299  *                                  ithread_mutexattr_t)
00300  *      int kind (one of ITHREAD_MUTEX_FAST_NP or ITHREAD_MUTEX_RECURSIVE_NP
00301  *                or ITHREAD_MUTEX_ERRORCHECK_NP)
00302  *  Returns:
00303  *      0 on success. Nonzero on failure.
00304  *      Returns EINVAL if the kind is not supported.
00305  *      See man page for pthread_mutexattr_setkind_np
00306  ****************************************************************************/
00307 #define ithread_mutexattr_setkind_np pthread_mutexattr_settype
00308 #define ithread_mutexattr_settype pthread_mutexattr_settype
00309
00310 /*****************************************************************************
00311  * Function: ithread_mutexattr_getkind_np
00312  *
00313  *  Description:
00314  *      Gets the mutex type in the attribute.
00315  *      Valid types are: ITHREAD_MUTEX_FAST_NP
00316  *                       ITHREAD_MUTEX_RECURSIVE_NP
00317  *                       ITHREAD_MUTEX_ERRORCHECK_NP
00318  *
00319  *  Parameters:
00320  *      ithread_mutexattr_t * attr (must be valid non NULL pointer to
00321  *                                  pthread_mutexattr_t)
00322  *      int *kind (one of ITHREAD_MUTEX_FAST_NP or ITHREAD_MUTEX_RECURSIVE_NP
00323  *                or ITHREAD_MUTEX_ERRORCHECK_NP)
00324  *  Returns:
00325  *      0 on success. Nonzero on failure.
00326  *      Always returns 0.
00327  *      See man page for pthread_mutexattr_getkind_np
00328  ****************************************************************************/
00329 #define ithread_mutexattr_getkind_np pthread_mutexattr_gettype
00330 #define ithread_mutexattr_gettype pthread_mutexattr_gettype
00331
00332 /*****************************************************************************
```

```
00333  * Function: ithread_mutex_init
00334  *
00335  *  Description:
00336  *      Initializes mutex.
00337  *      Must be called before use.
00338  *
00339  *  Parameters:
00340  *      ithread_mutex_t *mutex (must be valid non NULL pointer to
00341  *              pthread_mutex_t)
00342  *      const ithread_mutexattr_t *mutex_attr
00343  *
00344  * Returns:
00345  *      0 on success,
00346  *      Nonzero on failure.
00347  *      Always returns 0.
00348  *      See man page for pthread_mutex_init
00349  ********************************************************************************/
00350 #define ithread_mutex_init pthread_mutex_init
00351
00352 /********************************************************************************
00353  * Function: ithread_mutex_lock
00354  *
00355  * Description:
00356  *      Locks mutex.
00357  *
00358  * Parameters:
00359  *      ithread_mutex_t * mutex (must be valid non NULL pointer to
00360  *              pthread_mutex_t). The mutex must be initialized.
00361  *
00362  * Returns:
00363  *      0 on success, Nonzero on failure.
00364  *      Always returns 0.
00365  *      See man page for pthread_mutex_lock
00366  ********************************************************************************/
00367 #define ithread_mutex_lock pthread_mutex_lock
00368
00369 /********************************************************************************
00370  * Function: ithread_mutex_unlock
00371  *
00372  *  Description:
00373  *      Unlocks mutex.
00374  *
00375  *  Parameters:
00376  *      ithread_mutex_t * mutex (must be valid non NULL pointer to
00377  *              pthread_mutex_t). The mutex must be initialized.
00378  *
00379  *  Returns:
00380  *      0 on success, Nonzero on failure.
00381  *      Always returns 0.
00382  *      See man page for pthread_mutex_unlock
00383  ********************************************************************************/
00384 #define ithread_mutex_unlock pthread_mutex_unlock
00385
00386 /********************************************************************************
00387  * Function: ithread_mutex_destroy
00388  *
00389  *  Description:
00390  *      Releases any resources held by the mutex.
00391  *      Mutex can no longer be used after this call.
00392  *      Mutex is only destroyed when there are no longer any threads
00393  *      waiting on it. Mutex cannot be destroyed if it is locked.
00394  * Parameters:
00395  *      ithread_mutex_t * mutex (must be valid non NULL pointer to
00396  *              pthread_mutex_t). The mutex must be initialized.
00397  * Returns:
00398  *      0 on success.
00399  *      Nonzero on failure.
00400  *      Always returns 0.
00401  *      See man page for pthread_mutex_destroy
00402  ********************************************************************************/
00403 #define ithread_mutex_destroy pthread_mutex_destroy
00404
00405 /********************************************************************************
00406  * Function: ithread_rwlockattr_init
00407  *
00408  *  Description:
00409  *      Initializes a rwlock attribute variable to default values.
00410  *  Parameters:
00411  *      const ithread_rwlockattr_init *attr (must be valid non NULL pointer to
00412  *                                           pthread_rwlockattr_t)
00413  *  Returns:
00414  *      0 on success, Nonzero on failure.
00415  *      Always returns 0.
00416  *      See man page for pthread_rwlockattr_init
00417  ********************************************************************************/
00418 #if UPNP_USE_RWLOCK
00419         #define ithread_rwlockattr_init pthread_rwlockattr_init
```

```
00420 #endif /* UPNP_USE_RWLOCK */
00421
00422 /************************************************************************
00423  * Function: ithread_rwlockattr_destroy
00424  *
00425  *  Description:
00426  *      Releases any resources held by the rwlock attribute.
00427  *  Parameters:
00428  *      ithread_rwlockattr_t *attr (must be valid non NULL pointer to
00429  *                                  pthread_rwlockattr_t)
00430  *  Returns:
00431  *      0 on success, Nonzero on failure.
00432  *      Always returns 0.
00433  *      See man page for pthread_rwlockattr_destroy
00434  ************************************************************************/
00435 #if UPNP_USE_RWLOCK
00436         #define ithread_rwlockattr_destroy pthread_rwlockattr_destroy
00437 #endif /* UPNP_USE_RWLOCK */
00438
00439 /************************************************************************
00440  * Function: ithread_rwlockatttr_setpshared
00441  *
00442  *  Description:
00443  *      Sets the rwlock type in the attribute.
00444  *      Valid types are: ITHREAD_PROCESS_PRIVATE
00445  *                       ITHREAD_PROCESS_SHARED
00446  *
00447  *  Parameters:
00448  *      ithread_rwlockattr_t * attr (must be valid non NULL pointer to
00449  *                                   ithread_rwlockattr_t)
00450  *      int kind (one of ITHREAD_PROCESS_PRIVATE or ITHREAD_PROCESS_SHARED)
00451  *
00452  *  Returns:
00453  *      0 on success. Nonzero on failure.
00454  *      Returns EINVAL if the kind is not supported.
00455  *      See man page for pthread_rwlockattr_setkind_np
00456  ************************************************************************/
00457 #if UPNP_USE_RWLOCK
00458         #define ithread_rwlockatttr_setpshared pthread_rwlockatttr_setpshared
00459 #endif /* UPNP_USE_RWLOCK */
00460
00461 /************************************************************************
00462  * Function: ithread_rwlockatttr_getpshared
00463  *
00464  *  Description:
00465  *      Gets the rwlock type in the attribute.
00466  *      Valid types are: ITHREAD_PROCESS_PRIVATE
00467  *                       ITHREAD_PROCESS_SHARED
00468  *
00469  *  Parameters:
00470  *      ithread_rwlockattr_t * attr (must be valid non NULL pointer to
00471  *                                   pthread_rwlockattr_t)
00472  *      int *kind (one of ITHREAD_PROCESS_PRIVATE or ITHREAD_PROCESS_SHARED)
00473  *
00474  *  Returns:
00475  *      0 on success. Nonzero on failure.
00476  *      Always returns 0.
00477  *      See man page for pthread_rwlockatttr_getpshared
00478  ************************************************************************/
00479 #if UPNP_USE_RWLOCK
00480         #define ithread_rwlockatttr_getpshared pthread_rwlockatttr_getpshared
00481 #endif /* UPNP_USE_RWLOCK */
00482
00483 /************************************************************************
00484  * Function: ithread_rwlock_init
00485  *
00486  * Description:
00487  *      Initializes rwlock.
00488  *      Must be called before use.
00489  *
00490  * Parameters:
00491  *      ithread_rwlock_t *rwlock (must be valid non NULL pointer to
00492  *              pthread_rwlock_t) const ithread_rwlockattr_t *rwlock_attr
00493  *
00494  * Returns:
00495  *      0 on success
00496  *      Nonzero on failure.
00497  *      Always returns 0.
00498  *      See man page for pthread_rwlock_init
00499  ************************************************************************/
00500 #if UPNP_USE_RWLOCK
00501         #define ithread_rwlock_init pthread_rwlock_init
00502 #else
00503         /* Read-write locks aren't available: use mutex instead. */
00504         #define ithread_rwlock_init ithread_mutex_init
00505 #endif
00506
```

```
00507 /************************************************************************
00508  * Function: ithread_rwlock_rdlock
00509  *
00510  *  Description:
00511  *      Locks rwlock for reading.
00512  *  Parameters:
00513  *      ithread_rwlock_t *rwlock (must be valid non NULL pointer to
00514  *              pthread_rwlock_t). The rwlock must be initialized.
00515  *
00516  *  Returns:
00517  *      0 on success, Nonzero on failure.
00518  *      Always returns 0.
00519  *      See man page for pthread_rwlock_rdlock
00520  ************************************************************************/
00521 #if UPNP_USE_RWLOCK
00522         #define ithread_rwlock_rdlock pthread_rwlock_rdlock
00523 #else
00524         /* Read-write locks aren't available: use mutex instead. */
00525         #define ithread_rwlock_rdlock ithread_mutex_lock
00526 #endif /* UPNP_USE_RWLOCK */
00527
00528 /************************************************************************
00529  * Function: ithread_rwlock_wrlock
00530  *
00531  *  Description:
00532  *      Locks rwlock for writting.
00533  *  Parameters:
00534  *      ithread_rwlock_t *rwlock (must be valid non NULL pointer to
00535  *              pthread_rwlock_t) rwlock must be initialized.
00536  *
00537  *  Returns:
00538  *      0 on success, Nonzero on failure.
00539  *      Always returns 0.
00540  *      See man page for pthread_rwlock_wrlock
00541  ************************************************************************/
00542 #if UPNP_USE_RWLOCK
00543         #define ithread_rwlock_wrlock pthread_rwlock_wrlock
00544 #else
00545         /* Read-write locks aren't available: use mutex instead. */
00546         #define ithread_rwlock_wrlock ithread_mutex_lock
00547 #endif /* UPNP_USE_RWLOCK */
00548
00549 /************************************************************************
00550  * Function: ithread_rwlock_unlock
00551  *
00552  *  Description:
00553  *      Unlocks rwlock.
00554  *
00555  *  Parameters:
00556  *      ithread_rwlock_t *rwlock (must be valid non NULL pointer to
00557  *              pthread_rwlock_t) rwlock must be initialized.
00558  *
00559  *  Returns:
00560  *      0 on success, Nonzero on failure.
00561  *      Always returns 0.
00562  *      See man page for pthread_rwlock_unlock
00563  ************************************************************************/
00564 #if UPNP_USE_RWLOCK
00565         #define ithread_rwlock_unlock pthread_rwlock_unlock
00566 #else
00567         /* Read-write locks aren't available: use mutex instead. */
00568         #define ithread_rwlock_unlock ithread_mutex_unlock
00569 #endif /* UPNP_USE_RWLOCK */
00570
00571 /************************************************************************
00572  * Function: ithread_rwlock_destroy
00573  *
00574  *  Description:
00575  *      Releases any resources held by the rwlock.
00576  *      rwlock can no longer be used after this call.
00577  *      rwlock is only destroyed when there are no longer any threads
00578  *      waiting on it. rwlock cannot be destroyed if it is locked.
00579  *
00580  * Parameters:
00581  *      ithread_rwlock_t *rwlock (must be valid non NULL pointer to
00582  *              pthread_rwlock_t) rwlock must be initialized.
00583  *
00584  * Returns:
00585  *      0 on success.
00586  *      Nonzero on failure.
00587  *      Always returns 0.
00588  *      See man page for pthread_rwlock_destroy
00589  ************************************************************************/
00590 #if UPNP_USE_RWLOCK
00591         #define ithread_rwlock_destroy pthread_rwlock_destroy
00592 #else
00593         /* Read-write locks aren't available: use mutex instead. */
```

```
00594         #define ithread_rwlock_destroy ithread_mutex_destroy
00595 #endif /* UPNP_USE_RWLOCK */
00596
00597 /************************************************************************
00598  * Function: ithread_cond_init
00599  *
00600  *  Description:
00601  *      Initializes condition variable.
00602  *      Must be called before use.
00603  *  Parameters:
00604  *      ithread_cond_t *cond (must be valid non NULL pointer to pthread_cond_t)
00605  *      const ithread_condattr_t *cond_attr (ignored)
00606  *  Returns:
00607  *      0 on success, Nonzero on failure.
00608  *      See man page for pthread_cond_init
00609  ************************************************************************/
00610 #define ithread_cond_init pthread_cond_init
00611
00612 /************************************************************************
00613  * Function: ithread_cond_signal
00614  *
00615  *  Description:
00616  *      Wakes up exactly one thread waiting on condition.
00617  *      Associated mutex MUST be locked by thread before entering this call.
00618  *  Parameters:
00619  *      ithread_cond_t *cond (must be valid non NULL pointer to
00620  *      ithread_cond_t)
00621  *      cond must be initialized
00622  *  Returns:
00623  *      0 on success, Nonzero on failure.
00624  *      See man page for pthread_cond_signal
00625  ************************************************************************/
00626 #define ithread_cond_signal pthread_cond_signal
00627
00628 /************************************************************************
00629  * Function: ithread_cond_broadcast
00630  *
00631  *  Description:
00632  *      Wakes up all threads waiting on condition.
00633  *      Associated mutex MUST be locked by thread before entering this call.
00634  *  Parameters:
00635  *      ithread_cond_t *cond (must be valid non NULL pointer to
00636  *      ithread_cond_t)
00637  *      cond must be initialized
00638  *  Returns:
00639  *      0 on success, Nonzero on failure.
00640  *      See man page for pthread_cond_broadcast
00641  ************************************************************************/
00642 #define ithread_cond_broadcast pthread_cond_broadcast
00643
00644 /************************************************************************
00645  * Function: ithread_cond_wait
00646  *
00647  *  Description:
00648  *      Atomically releases mutex and waits on condition.
00649  *      Associated mutex MUST be locked by thread before entering this call.
00650  *      Mutex is reacquired when call returns.
00651  *  Parameters:
00652  *      ithread_cond_t *cond (must be valid non NULL pointer to
00653  *      ithread_cond_t)
00654  *      cond must be initialized
00655  *      ithread_mutex_t *mutex (must be valid non NULL pointer to
00656  *      ithread_mutex_t)
00657  *      Mutex must be locked.
00658  *  Returns:
00659  *      0 on success, Nonzero on failure.
00660  *      See man page for pthread_cond_wait
00661  ************************************************************************/
00662 #define ithread_cond_wait pthread_cond_wait
00663
00664 /************************************************************************
00665  * Function: pthread_cond_timedwait
00666  *
00667  * Description:
00668  *      Atomically releases the associated mutex and waits on the condition.
00669  *      If the condition is not signaled in the specified time than the call
00670  *      times out and returns. Associated mutex MUST be locked by thread before
00671  *      entering this call. Mutex is reacquired when call returns.
00672  * Parameters:
00673  *      ithread_cond_t *cond (must be valid non NULL pointer to ithread_cond_t)
00674  *              cond must be initialized
00675  *      ithread_mutex_t *mutex (must be valid non NULL pointer to
00676  *              ithread_mutex_t) Mutex must be locked.
00677  *      const struct timespec *abstime(absolute time, measured from Jan 1, 1970)
00678  * Returns:
00679  *      0 on success.
00680  *      ETIMEDOUT on timeout.
```

```
00681  *      Nonzero on failure.
00682  *      See man page for pthread_cond_timedwait
00683  ******************************************************************************/
00684
00685 #define ithread_cond_timedwait pthread_cond_timedwait
00686
00687 /******************************************************************************
00688  * Function: ithread_cond_destroy
00689  *
00690  *  Description:
00691  *      Releases any resources held by the condition variable.
00692  *              Condition variable can no longer be used after this call.
00693  *  Parameters:
00694  *      ithread_cond_t *cond (must be valid non NULL pointer to
00695  *      ithread_cond_t)
00696  *      cond must be initialized.
00697  *  Returns:
00698  *      0 on success. Nonzero on failure.
00699  *      See man page for pthread_cond_destroy
00700  ******************************************************************************/
00701 #define ithread_cond_destroy pthread_cond_destroy
00702
00703 /******************************************************************************
00704  * Function: ithread_attr_init
00705  *
00706  *  Description:
00707  *      Initialises thread attribute object.
00708  *  Parameters:
00709  *      ithread_attr_t *attr (must be valid non NULL pointer to
00710  *      ithread_attr_t)
00711  *  Returns:
00712  *      0 on success. Nonzero on failure.
00713  *      See man page for pthread_attr_init
00714  ******************************************************************************/
00715 #define ithread_attr_init pthread_attr_init
00716
00717 /******************************************************************************
00718  * Function: ithread_attr_destroy
00719  *
00720  *  Description:
00721  *      Destroys thread attribute object.
00722  *  Parameters:
00723  *      ithread_attr_t *attr (must be valid non NULL pointer to
00724  *      ithread_attr_t)
00725  *  Returns:
00726  *      0 on success. Nonzero on failure.
00727  *      See man page for pthread_attr_destroy
00728  ******************************************************************************/
00729 #define ithread_attr_destroy pthread_attr_destroy
00730
00731 /******************************************************************************
00732  * Function: ithread_attr_setstacksize
00733  *
00734  *  Description:
00735  *      Sets stack size of a thread attribute object.
00736  *  Parameters:
00737  *      ithread_attr_t *attr (must be valid non NULL pointer to
00738  *      ithread_attr_t)
00739  *      size_t stacksize (value of stacksize must be greater than
00740  *      ITHREAD_STACK_MIN and lower than system-imposed limits
00741  *  Returns:
00742  *      0 on success. Nonzero on failure.
00743  *      See man page for pthread_attr_setstacksize
00744  ******************************************************************************/
00745 #define ithread_attr_setstacksize pthread_attr_setstacksize
00746
00747 /******************************************************************************
00748  * Function: ithread_attr_setdetachstate
00749  *
00750  *  Description:
00751  *      Sets detach state of a thread attribute object.
00752  *  Parameters:
00753  *      ithread_attr_t *attr (must be valid non NULL pointer to
00754  *      ithread_attr_t)
00755  *      int detachstate (value of detachstate must be ITHREAD_CREATE_DETACHED
00756  *      or ITHREAD_CREATE_JOINABLE)
00757  *  Returns:
00758  *      0 on success. Nonzero on failure.
00759  *      See man page for pthread_attr_setdetachstate
00760  ******************************************************************************/
00761 #define ithread_attr_setdetachstate pthread_attr_setdetachstate
00762
00763 /******************************************************************************
00764  * Function: ithread_create
00765  *
00766  *  Description:
00767  *              Creates a thread with the given start routine
```

```
00768  *      and argument.
00769  *  Parameters:
00770  *      ithread_t * thread (must be valid non NULL pointer to pthread_t)
00771  *      ithread_attr_t *attr
00772  *      void * (start_routine) (void *arg) (start routine)
00773  *      void * arg - argument.
00774  *  Returns:
00775  *      0 on success. Nonzero on failure.
00776  *          Returns EAGAIN if a new thread can not be created.
00777  *      Returns EINVAL if there is a problem with the arguments.
00778  *      See man page fore pthread_create
00779  ********************************************************************************/
00780 #define ithread_create pthread_create
00781
00782 /********************************************************************************
00783  * Function: ithread_cancel
00784  *
00785  *  Description:
00786  *              Cancels a thread.
00787  *  Parameters:
00788  *      ithread_t * thread (must be valid non NULL pointer to ithread_t)
00789  *  Returns:
00790  *      0 on success. Nonzero on failure.
00791  *      See man page for pthread_cancel
00792  ********************************************************************************/
00793 #define ithread_cancel pthread_cancel
00794
00795 /********************************************************************************
00796  * Function: ithread_exit
00797  *
00798  *  Description:
00799  *              Returns a return code from a thread.
00800  *      Implicitly called when the start routine returns.
00801  *  Parameters:
00802  *      void  * return_code return code to return
00803  *      See man page for pthread_exit
00804  ********************************************************************************/
00805 #define ithread_exit pthread_exit
00806
00807 /********************************************************************************
00808  * Function: ithread_get_current_thread_id
00809  *
00810  *  Description:
00811  *              Returns the handle of the currently running thread.
00812  *  Returns:
00813  *              The handle of the currently running thread.
00814  *              See man page for pthread_self
00815  ********************************************************************************/
00816 #define ithread_get_current_thread_id pthread_self
00817
00818 /********************************************************************************
00819  * Function: ithread_self
00820  *
00821  *  Description:
00822  *              Returns the handle of the currently running thread.
00823  *  Returns:
00824  *              The handle of the currently running thread.
00825  *              See man page for pthread_self
00826  ********************************************************************************/
00827 #define ithread_self pthread_self
00828
00829 /********************************************************************************
00830  * Function: ithread_detach
00831  *
00832  *  Description:
00833  *              Makes a thread's resources reclaimed immediately
00834  *            after it finishes
00835  *            execution.
00836  *  Returns:
00837  *              0 on success, Nonzero on failure.
00838  *      See man page for pthread_detach
00839  ********************************************************************************/
00840 #define ithread_detach pthread_detach
00841
00842 /********************************************************************************
00843  * Function: ithread_join
00844  *
00845  *  Description:
00846  *              Suspends the currently running thread until the
00847  * specified thread
00848  *      has finished.
00849  *      Returns the return code of the thread, or ITHREAD_CANCELED
00850  *      if the thread has been canceled.
00851  *  Parameters:
00852  *      ithread_t *thread (valid non null thread identifier)
00853  *      void ** return (space for return code)
00854  *  Returns:
```

```
00855  *              0 on success, Nonzero on failure.
00856  *     See man page for pthread_join
00857  ************************************************************************/
00858 #define ithread_join pthread_join
00859
00860 /************************************************************************
00861  * Function: isleep
00862  *
00863  *  Description:
00864  *              Suspends the currently running thread for the specified number
00865  *      of seconds
00866  *      Always returns 0.
00867  *  Parameters:
00868  *      unsigned int seconds - number of seconds to sleep.
00869  *  Returns:
00870  *              0 on success, Nonzero on failure.
00871  *              See man page for sleep (man 3 sleep)
00872  ************************************************************************/
00873 #ifdef _WIN32
00874         #define isleep(x) Sleep((x)*1000)
00875 #else
00876         #define isleep sleep
00877 #endif
00878
00879 /************************************************************************
00880  * Function: isleep
00881  *
00882  *  Description:
00883  *              Suspends the currently running thread for the specified number
00884  *      of milliseconds
00885  *      Always returns 0.
00886  *  Parameters:
00887  *      unsigned int milliseconds - number of milliseconds to sleep.
00888  *  Returns:
00889  *              0 on success, Nonzero on failure.
00890  *              See man page for sleep (man 3 sleep)
00891  ************************************************************************/
00892 #ifdef _WIN32
00893         #define imillisleep Sleep
00894 #else
00895         #if _POSIX_C_SOURCE < 200809L
00896                 #define imillisleep(x) usleep(1000 * x)
00897         #else
00898                 #define imillisleep(x) \
00899                         do { \
00900                                 const struct timespec req = { \
00901                                         0, x * 1000 * 1000}; \
00902                                 nanosleep(&req, NULL); \
00903                         } while (0)
00904         #endif
00905 #endif
00906
00907 #if !defined(PTHREAD_MUTEX_RECURSIVE) && !defined(__DragonFly__) && \
00908         !defined(UPNP_USE_MSVCPP)
00909 /* !defined(UPNP_USE_MSVCPP) should probably also have pthreads version check -
00910  * but it's not clear if that is possible */
00911 /* NK: Added for satisfying the gcc compiler */
00912 UPNP_EXPORT_SPEC int pthread_mutexattr_setkind_np(
00913         pthread_mutexattr_t *attr, int kind);
00914 #endif
00915
00916 #ifdef __cplusplus
00917 }
00918 #endif
00919
00920 #endif /* ITHREAD_H */
```

9.28 list.h

```
00001 #ifndef _UPNP_LIST_H_
00002 #define _UPNP_LIST_H_
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  * Copyright (C) 2011-2012 France Telecom All rights reserved.
00009  *
00010  * Redistribution and use in source and binary forms, with or without
00011  * modification, are permitted provided that the following conditions are met:
00012  *
00013  * * Redistributions of source code must retain the above copyright notice,
00014  * this list of conditions and the following disclaimer.
```

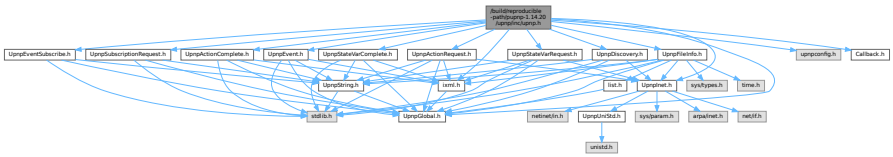
```
00015  * * Redistributions in binary form must reproduce the above copyright notice,
00016  * this list of conditions and the following disclaimer in the documentation
00017  * and/or other materials provided with the distribution.
00018  * * Neither name of Intel Corporation nor the names of its contributors
00019  * may be used to endorse or promote products derived from this software
00020  * without specific prior written permission.
00021  *
00022  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00026  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00027  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00028  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00029  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00030  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00031  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00032  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  ******************************************************************************/
00035
00041 #include "UpnpGlobal.h"
00042
00043 #ifdef __cplusplus
00044 extern "C" {
00045 #endif /* __cplusplus */
00046
00051 typedef struct UpnpListHead
00052 {
00053         struct UpnpListHead *next, *prev;
00054 } UpnpListHead;
00055
00057 typedef UpnpListHead *UpnpListIter;
00058
00060 UPNP_EXPORT_SPEC void UpnpListInit(UpnpListHead *list);
00061
00064 UPNP_EXPORT_SPEC UpnpListIter UpnpListBegin(UpnpListHead *list);
00065
00067 UPNP_EXPORT_SPEC UpnpListIter UpnpListEnd(UpnpListHead *list);
00068
00070 UPNP_EXPORT_SPEC UpnpListIter UpnpListNext(
00071         UpnpListHead *list, UpnpListIter pos);
00072
00074 UPNP_EXPORT_SPEC UpnpListIter UpnpListInsert(
00075         UpnpListHead *list, UpnpListIter pos, UpnpListHead *elt);
00076
00078 UPNP_EXPORT_SPEC UpnpListIter UpnpListErase(
00079         UpnpListHead *list, UpnpListIter pos);
00080
00081 #ifdef __cplusplus
00082 }
00083 #endif /* __cplusplus */
00084
00085 #endif /* _UPNPLISTH_ */
```

9.29 /build/reproducible-path/pupnp-1.14.20/upnp/inc/upnp.h File Reference

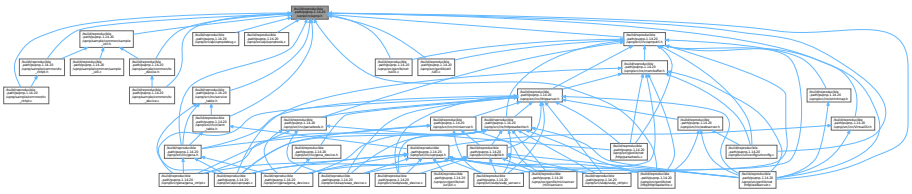
```
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include "ixml.h"
#include "upnpconfig.h"
#include "UpnpActionComplete.h"
#include "UpnpActionRequest.h"
#include "UpnpDiscovery.h"
#include "UpnpEvent.h"
#include "UpnpEventSubscribe.h"
#include "UpnpFileInfo.h"
#include "UpnpStateVarComplete.h"
#include "UpnpStateVarRequest.h"
#include "UpnpSubscriptionRequest.h"
```

```
#include "Callback.h"
```

Include dependency graph for upnp.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **LINE_SIZE** (size_t)180
- #define **NAME_SIZE** (size_t)256
- #define **MNFT_NAME_SIZE** 64
- #define **MODL_NAME_SIZE** 32
- #define **SERL_NUMR_SIZE** 64
- #define **MODL_DESC_SIZE** 64
- #define **UPNP_INFINITE** -1
- #define **UPNP_USING_CHUNKED** -3
- #define **UPNP_UNTIL_CLOSE** -4

Error codes

The functions in the SDK API can return a variety of error codes to describe problems encountered during execution. This section lists the error codes and provides a brief description of what each error code means. Refer to the documentation for each function for a description of what an error code means in that context.

- #define UPNP_E_SUCCESS 0

  The operation completed successfully.
- #define UPNP_E_INVALID_HANDLE -100

  The handle passed to a function is not a recognized as a valid handle.
- #define UPNP_E_INVALID_PARAM -101

  One or more of the parameters passed to the function is not valid.
- #define UPNP_E_OUTOF_HANDLE -102

  The SDK does not have any more space for additional handles.
- #define **UPNP_E_OUTOF_CONTEXT** -103
- #define UPNP_E_OUTOF_MEMORY -104

  Not enough resources are currently available to complete the operation.
- #define UPNP_E_INIT -105

  The SDK has already been initialized.
- #define **UPNP_E_BUFFER_TOO_SMALL** -106
- #define UPNP_E_INVALID_DESC -107

  *The description document passed to **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2** **UpnpRegister↩ RootDevice3** or **UpnpRegisterRootDevice4** is invalid.*

- #define UPNP_E_INVALID_URL -108

  An URL passed into the function is invalid.
- #define **UPNP_E_INVALID_SID** -109
- #define **UPNP_E_INVALID_DEVICE** -110
- #define UPNP_E_INVALID_SERVICE -111

  The device ID/service ID pair does not refer to a valid service.
- #define UPNP_E_BAD_RESPONSE -113

  The response received from the remote side of a connection is not correct for the protocol.
- #define **UPNP_E_BAD_REQUEST** -114
- #define UPNP_E_INVALID_ACTION -115

  The SOAP action message is invalid.
- #define UPNP_E_FINISH -116

  ***UpnpInit2** has not been called, or **UpnpFinish** has already been called.*
- #define UPNP_E_INIT_FAILED -117

  ***UpnpInit2** cannot complete.*
- #define UPNP_E_URL_TOO_BIG -118

  The URL passed into a function is too long.
- #define UPNP_E_BAD_HTTPMSG -119

  The HTTP message contains invalid message headers.
- #define UPNP_E_ALREADY_REGISTERED -120

  A client or a device is already registered.
- #define UPNP_E_INVALID_INTERFACE -121

  *The interface provided to **UpnpInit2** is unknown or does not have a valid IPv4 or IPv6 address configured.*
- #define UPNP_E_NETWORK_ERROR -200

  A network error occurred.
- #define UPNP_E_SOCKET_WRITE -201

  An error happened while writing to a socket.
- #define UPNP_E_SOCKET_READ -202

  An error happened while reading from a socket.
- #define UPNP_E_SOCKET_BIND -203

  The SDK had a problem binding a socket to a network interface.
- #define UPNP_E_SOCKET_CONNECT -204

  The SDK had a problem connecting to a remote host.
- #define UPNP_E_OUTOF_SOCKET -205

  The SDK cannot create any more sockets.
- #define UPNP_E_LISTEN -206

  The SDK had a problem setting the socket to listen for incoming connections.
- #define UPNP_E_TIMEDOUT -207

  Too much time elapsed before the required number of bytes were sent or received over a socket.
- #define UPNP_E_SOCKET_ERROR -208

  Generic socket error code for conditions not covered by other error codes.
- #define **UPNP_E_FILE_WRITE_ERROR** -209
- #define UPNP_E_CANCELED -210

  The operation was canceled.
- #define **UPNP_E_EVENT_PROTOCOL** -300
- #define UPNP_E_SUBSCRIBE_UNACCEPTED -301

  A subscription request was rejected from the remote side.
- #define UPNP_E_UNSUBSCRIBE_UNACCEPTED -302

  An unsubscribe request was rejected from the remote side.
- #define UPNP_E_NOTIFY_UNACCEPTED -303

  The remote host did not accept the notify sent from the local device.
- #define UPNP_E_INVALID_ARGUMENT -501

  One or more of the parameters passed to a function is invalid.
- #define UPNP_E_FILE_NOT_FOUND -502

  The filename passed to one of the device registration functions was not found or was not accessible.
- #define UPNP_E_FILE_READ_ERROR -503

  An error happened while reading a file.
- #define UPNP_E_EXT_NOT_XML -504

  *The file name of the description document passed to **UpnpRegisterRootDevice2** does not end in ".xml".*

- #define **UPNP_E_NO_WEB_SERVER** -505
- #define **UPNP_E_OUTOF_BOUNDS** -506
- #define UPNP_E_NOT_FOUND -507

  The response to a SOAP request did not contain the required XML constructs.
- #define UPNP_E_INTERNAL_ERROR -911

  Generic error code for internal conditions not covered by other error codes.
- #define **UPNP_SOAP_E_INVALID_ACTION** 401
- #define **UPNP_SOAP_E_INVALID_ARGS** 402
- #define **UPNP_SOAP_E_OUT_OF_SYNC** 403
- #define **UPNP_SOAP_E_INVALID_VAR** 404
- #define **UPNP_SOAP_E_ACTION_FAILED** 501

Functions

Initialization and Registration

- int UpnpInit2 (const char *lfName, unsigned short DestPort)

  Initializes the Linux SDK for UPnP Devices (IPv4 or IPv6).
- int UpnpFinish (void)

  Initializes the OpenSSL library, and the OpenSSL context for use with pupnp.
- unsigned short UpnpGetServerPort (void)

  Returns the internal server IPv4 UPnP listening port.
- unsigned short UpnpGetServerPort6 (void)

  Returns the internal server IPv6 link-local (LLA) UPnP listening port.
- unsigned short UpnpGetServerUlaGuaPort6 (void)

  Returns the internal server IPv6 ULA or GUA UPnP listening port.
- char * UpnpGetServerIpAddress (void)

  Returns the local IPv4 listening ip address.
- char * UpnpGetServerIp6Address (void)

  Returns the IPv6 link-local listening ip address.
- char * UpnpGetServerUlaGuaIp6Address (void)

  Returns the IPv6 unique-local or globally-unique listening ip address.
- int UpnpRegisterRootDevice (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd)

  Registers a device application with the UPnP Library.
- int UpnpRegisterRootDevice2 (Upnp_DescType descriptionType, const char *description, size_t buffer↩Len, int config_baseURL, Upnp_FunPtr Fun, const void *Cookie, UpnpDevice_Handle *Hnd)

  *Registers a device application with the UPnP Library. Similar to **UpnpRegisterRootDevice**, except that it also allows the description document to be specified as a file or a memory buffer.*
- int UpnpRegisterRootDevice3 (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd, int AddressFamily)

  Registers a device application for a specific address family with the UPnP library.
- int UpnpRegisterRootDevice4 (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd, int AddressFamily, const char *LowerDescUrl)

  Registers a device application for a specific address family with the UPnP library. This function can also be used to specify a dedicated description URL to be returned for legacy CPs.
- int UpnpUnRegisterRootDevice (UpnpDevice_Handle Hnd)

  *Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **Upnp**↩**RegisterRootDevice3** or **UpnpRegisterRootDevice4**.*
- int UpnpUnRegisterRootDeviceLowPower (UpnpDevice_Handle Hnd, int PowerState, int SleepPeriod, int RegistrationState)

  *Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **Upnp**↩**RegisterRootDevice3** or **UpnpRegisterRootDevice4**.*
- int UpnpRegisterClient (Upnp_FunPtr Callback, const void *Cookie, UpnpClient_Handle *Hnd)

  Registers a control point application with the UPnP Library.
- int UpnpUnRegisterClient (UpnpClient_Handle Hnd)

  Unregisters a control point application, unsubscribing all active subscriptions.
- int UpnpSetContentLength (UpnpClient_Handle Hnd, size_t contentLength)
- int UpnpSetMaxContentLength (size_t contentLength)

Sets the maximum content-length that the SDK will process on an incoming SOAP requests or responses.

Discovery

- int UpnpSearchAsync (UpnpClient_Handle Hnd, int Mx, const char *TTarget_constarget_const, const void *Cookie_const)
  Searches for devices matching the given search target.
- int UpnpSendAdvertisement (UpnpDevice_Handle Hnd, int Exp)
  Sends out the discovery announcements for all devices and services for a device.
- int UpnpSendAdvertisementLowPower (UpnpDevice_Handle Hnd, int Exp, int PowerState, int SleepPeriod, int RegistrationState)
  Sends out the discovery announcements for all devices and services for a device.

Control

- int UpnpGetServiceVarStatus (UpnpClient_Handle Hnd, const char *ActionURL, const char *VarName, DOMString *StVarVal)
  Queries the state of a state variable of a service on another device.
- int UpnpGetServiceVarStatusAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *VarName, Upnp_FunPtr Fun, const void *Cookie)
  Queries the state of a variable of a service, generating a callback when the operation is complete.
- int UpnpSendAction (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Action, IXML_Document **RespNode)
  Sends a message to change a state variable in a service.
- int UpnpSendActionEx (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Header, IXML_Document *Action, IXML_Document **RespNode)
  Sends a message to change a state variable in a service.
- int UpnpSendActionAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Action, Upnp_FunPtr Fun, const void *Cookie)
  Sends a message to change a state variable in a service, generating a callback when the operation is complete.
- int UpnpSendActionExAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Header, IXML_Document *Action, Upnp_FunPtr Fun, const void *Cookie)
  Sends a message to change a state variable in a service, generating a callback when the operation is complete.

Eventing

- int UpnpAcceptSubscription (UpnpDevice_Handle Hnd, const char *DevID, const char *ServID, const char **VarName, const char **NewVal, int cVariables, const Upnp_SID SubsId)
  Accepts a subscription request and sends out the current state of the eventable variables for a service.
- int UpnpAcceptSubscriptionExt (UpnpDevice_Handle Hnd, const char *DevID, const char *ServID, IXML_Document *PropSet, const Upnp_SID SubsId)
  *Similar to **UpnpAcceptSubscription()** except that it takes a DOM document for the variables to event rather than an array of strings.*
- int UpnpNotify (UpnpDevice_Handle, const char *DevID, const char *ServID, const char **VarName, const char **NewVal, int cVariables)
  Sends out an event change notification to all control points subscribed to a particular service.
- int UpnpNotifyExt (UpnpDevice_Handle, const char *DevID, const char *ServID, IXML_Document *PropSet)
  *Similar to **UpnpNotify** except that it takes a DOM document for the event rather than an array of strings.*
- int UpnpRenewSubscription (UpnpClient_Handle Hnd, int *TimeOut, const Upnp_SID SubsId)
  Renews a subscription that is about to expire.
- int UpnpRenewSubscriptionAsync (UpnpClient_Handle Hnd, int TimeOut, Upnp_SID SubsId, Upnp_FunPtr Fun, const void *Cookie)
  Renews a subscription that is about to expire, generating a callback when the operation is complete.
- int UpnpSetMaxSubscriptions (UpnpDevice_Handle Hnd, int MaxSubscriptions)
  Sets the maximum number of subscriptions accepted per service.

- int UpnpSetMaxSubscriptionTimeOut (UpnpDevice_Handle Hnd, int MaxSubscriptionTimeOut)

  Sets the maximum time-out accepted for a subscription request or renewal.
- int UpnpSubscribe (UpnpClient_Handle Hnd, const char *PublisherUrl, int *TimeOut, Upnp_SID SubsId)

  Registers a control point to receive event notifications from another device.
- int UpnpSubscribeAsync (UpnpClient_Handle Hnd, const char *PublisherUrl, int TimeOut, Upnp_FunPtr Fun, const void *Cookie)

  *Performs the same operation as **UpnpSubscribe**, but returns immediately and calls the registered callback function when the operation is complete.*
- int UpnpUnSubscribe (UpnpClient_Handle Hnd, const Upnp_SID SubsId)

  *Removes the subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**.*
- int UpnpUnSubscribeAsync (UpnpClient_Handle Hnd, Upnp_SID SubsId, Upnp_FunPtr Fun, const void *Cookie)

  *Removes a subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**, generating a callback when the operation is complete.*

Constants and Types

- enum **UpnpOpenFileMode** { **UPNP_READ** , **UPNP_WRITE** }
- enum Upnp_SType_e { UPNP_S_ALL , UPNP_S_ROOT , UPNP_S_DEVICE , UPNP_S_SERVICE }

  Represents the different types of searches that can be performed using the SDK for UPnP Devices API.
- enum Upnp_DescType_e { UPNPREG_URL_DESC , UPNPREG_FILENAME_DESC , UPNPREG_BUF_DESC }

  *Specifies the type of description in **UpnpRegisterRootDevice2**.*
- typedef int UpnpClient_Handle

  *Returned when a control point application registers with **UpnpRegisterClient**.*
- typedef int UpnpDevice_Handle

  *Returned when a device application registers with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **UpnpRegisterRootDevice3** or **UpnpRegisterRootDevice4**.*
- typedef char Upnp_SID[44]

  Holds the subscription identifier for a subscription between a client and a device.
- typedef enum Upnp_SType_e **Upnp_SType**
- typedef enum Upnp_DescType_e **Upnp_DescType**

Control Point HTTP API

- enum Upnp_HttpMethod_e {
  UPNP_HTTPMETHOD_PUT = 0 , **UPNP_HTTPMETHOD_DELETE** = 1 , **UPNP_HTTPMETHOD_GET** = 2 ,
  UPNP_HTTPMETHOD_HEAD = 3 ,
  UPNP_HTTPMETHOD_POST = 4 }

  Different HTTP methods.
- typedef enum Upnp_HttpMethod_e **Upnp_HttpMethod**
- int UpnpDownloadUrlItem (const char *url, char **outBuf, char *contentType)

  Downloads a file specified in a URL.
- int UpnpOpenHttpGet (const char *url, void **handle, char **contentType, int *contentLength, int *http↩ Status, int timeout)

  Gets a file specified in a URL.
- int UpnpOpenHttpGetProxy (const char *url, const char *proxy_str, void **handle, char **contentType, int *contentLength, int *httpStatus, int timeout)

  Gets a file specified in a URL through the specified proxy.
- int UpnpOpenHttpGetEx (const char *url, void **handle, char **contentType, int *contentLength, int *http↩ Status, int lowRange, int highRange, int timeout)

  Gets specified number of bytes from a file specified in the URL.

- int UpnpReadHttpGet (void *handle, char *buf, size_t *size, int timeout)

  Gets specified number of bytes from a file specified in a URL.
- int UpnpHttpGetProgress (void *handle, size_t *length, size_t *total)

  Retrieve progress information of a http-get transfer.
- int UpnpCancelHttpGet (void *handle)

  *Set the cancel flag of the **handle** parameter.*
- int UpnpCloseHttpGet (void *handle)

  *Closes the connection and frees memory that was allocated for the **handle** parameter.*
- int UpnpOpenHttpPost (const char *url, void **handle, const char *contentType, int contentLength, int timeout)

  Makes an HTTP POST request message, opens a connection to the server and sends the POST request to the server if the connection to the server succeeds.
- int UpnpWriteHttpPost (void *handle, char *buf, size_t *size, int timeout)

  *Sends a request to a server to copy the contents of a buffer to the URI specified in the **UpnpOpenHttpPost** call.*
- int UpnpCloseHttpPost (void *handle, int *httpStatus, int timeout)

  *Sends and receives any pending data, closes the connection with the server, and frees memory allocated during the **UpnpOpenHttpPost** call.*
- int UpnpOpenHttpConnection (const char *url, void **handle, int timeout)

  Opens a connection to the server.
- int UpnpMakeHttpRequest (Upnp_HttpMethod method, const char *url, void *handle, UpnpString *headers, const char *contentType, int contentLength, int timeout)

  *Makes a HTTP request using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpWriteHttpRequest (void *handle, char *buf, size_t *size, int timeout)

  *Writes the content of a HTTP request initiated by a **UpnpMakeHttpRequest** call. The end of the content should be indicated by a call to **UpnpEndHttpRequest**.*
- int UpnpEndHttpRequest (void *handle, int timeout)

  *Indicates the end of a HTTP request previously made by **UpnpMakeHttpRequest**.*
- int UpnpGetHttpResponse (void *handle, UpnpString *headers, char **contentType, int *contentLength, int *httpStatus, int timeout)

  *Gets the response from the server using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpReadHttpResponse (void *handle, char *buf, size_t *size, int timeout)

  *Reads the content of a response using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpCloseHttpConnection (void *handle)

  *Closes the connection created with **UpnpOpenHttpConnection** and frees any memory associated with the connection.*
- int UpnpDownloadXmlDoc (const char *url, IXML_Document **xmlDoc)

  Downloads an XML document specified in a URL.

Web Server API

- typedef void * UpnpWebFileHandle

  The type of handle returned by the web server for open requests.
- typedef int(* VDCallback_GetInfo) (const char *filename, UpnpFileInfo *info, const void *cookie, const void **request_cookie)

  Get-info callback function prototype.
- typedef UpnpWebFileHandle(* VDCallback_Open) (const char *filename, enum UpnpOpenFileMode Mode, const void *cookie, const void *request_cookie)

  Open callback function prototype.
- typedef int(* VDCallback_Read) (UpnpWebFileHandle fileHnd, char *buf, size_t buflen, const void *cookie, const void *request_cookie)

  Read callback function prototype.

- typedef int(∗ VDCallback_Write) (UpnpWebFileHandle fileHnd, char ∗buf, size_t buflen, const void ∗cookie, const void ∗request_cookie)

  Write callback function prototype.
- typedef int(∗ VDCallback_Seek) (UpnpWebFileHandle fileHnd, off_t offset, int origin, const void ∗cookie, const void ∗request_cookie)

  Seek callback function prototype.
- typedef int(∗ VDCallback_Close) (UpnpWebFileHandle fileHnd, const void ∗cookie, const void ∗request_↩ cookie)

  Close callback function prototype.
- typedef int(∗ **WebCallback_HostValidate**) (const char ∗hostname, void ∗cookie)
- int UpnpSetWebServerRootDir (const char ∗rootDir)

  Sets the document root directory for the internal web server.
- int UpnpVirtualDir_set_GetInfoCallback (VDCallback_GetInfo callback)

  Sets the get_info callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_OpenCallback (VDCallback_Open callback)

  Sets the open callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_ReadCallback (VDCallback_Read callback)

  Sets the read callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_WriteCallback (VDCallback_Write callback)

  Sets the write callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_SeekCallback (VDCallback_Seek callback)

  Sets the seek callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_CloseCallback (VDCallback_Close callback)

  Sets the close callback function to be used to access a virtual directory.
- int UpnpEnableWebserver (int enable)

  Enables or disables the webserver.
- int UpnpIsWebserverEnabled (void)

  Returns `1` if the webserver is enabled, or `0` if it is not.
- void **UpnpSetHostValidateCallback** (WebCallback_HostValidate callback, void ∗cookie)
- void **UpnpSetAllowLiteralHostRedirection** (int enable)
- int UpnpSetWebServerCorsString (const char ∗corsString)

  Assign the Access-Control-Allow-Origin specfied by the input const char∗ cors_string parameterto the global CORS string.
- int UpnpAddVirtualDir (const char ∗dirName, const void ∗cookie, const void ∗∗oldcookie)

  Adds a virtual directory mapping.
- int UpnpRemoveVirtualDir (const char ∗dirName)

  *Removes a virtual directory mapping made with **UpnpAddVirtualDir**.*
- void UpnpRemoveAllVirtualDirs (void)

  Removes all virtual directory mappings.

9.30 upnp.h

Go to the documentation of this file.

```
00001 #ifndef UPNP_H
00002 #define UPNP_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  * Copyright (C) 2011-2012 France Telecom All rights reserved.
00009  *
00010  * Redistribution and use in source and binary forms, with or without
00011  * modification, are permitted provided that the following conditions are met:
00012  *
```

```
00013  * * Redistributions of source code must retain the above copyright notice,
00014  * this list of conditions and the following disclaimer.
00015  * * Redistributions in binary form must reproduce the above copyright notice,
00016  * this list of conditions and the following disclaimer in the documentation
00017  * and/or other materials provided with the distribution.
00018  * * Neither name of Intel Corporation nor the names of its contributors
00019  * may be used to endorse or promote products derived from this software
00020  * without specific prior written permission.
00021  *
00022  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00026  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00027  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00028  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00029  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00030  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00031  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00032  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  ******************************************************************************/
00035
00044 #include "UpnpGlobal.h"
00045 #include "UpnpInet.h"
00046 #include "ixml.h"
00047 #include "upnpconfig.h"
00048
00049 /*
00050  * \todo Document the exact reason of these include files and solve this
00051  * include mess in an include file like UpnpTime.h
00052  */
00053 #ifdef _WIN32
00054         #include <time.h>
00055         #if defined(UPNP_USE_MSVCPP) || defined(UPNP_USE_BCBPP)
00056                 #include <sys/types.h> /* needed for off_t */
00057         #endif
00058 #elif (defined(BSD) && BSD >= 199306)
00059         #include <time.h>
00060 #else
00061         /* Other systems ??? */
00062 #endif
00063
00064 #ifdef UPNP_ENABLE_OPEN_SSL
00065         #include <openssl/ssl.h>
00066 #endif
00067
00068 #define LINE_SIZE (size_t)180
00069 #define NAME_SIZE (size_t)256
00070 #define MNFT_NAME_SIZE 64
00071 #define MODL_NAME_SIZE 32
00072 #define SERL_NUMR_SIZE 64
00073 #define MODL_DESC_SIZE 64
00074 #define UPNP_INFINITE -1
00075 #define UPNP_USING_CHUNKED -3
00076 #define UPNP_UNTIL_CLOSE -4
00077
00097 #define UPNP_E_SUCCESS 0
00098
00102 #define UPNP_E_INVALID_HANDLE -100
00103
00110 #define UPNP_E_INVALID_PARAM -101
00111
00117 #define UPNP_E_OUTOF_HANDLE -102
00118
00119 #define UPNP_E_OUTOF_CONTEXT -103
00120
00127 #define UPNP_E_OUTOF_MEMORY -104
00128
00135 #define UPNP_E_INIT -105
00136
00137 #define UPNP_E_BUFFER_TOO_SMALL -106
00138
00144 #define UPNP_E_INVALID_DESC -107
00145
00153 #define UPNP_E_INVALID_URL -108
00154
00155 #define UPNP_E_INVALID_SID -109
00156
00157 #define UPNP_E_INVALID_DEVICE -110
00158
00165 #define UPNP_E_INVALID_SERVICE -111
00166
00173 #define UPNP_E_BAD_RESPONSE -113
00174
00175 #define UPNP_E_BAD_REQUEST -114
00176
```

```
00183 #define UPNP_E_INVALID_ACTION -115
00184
00191 #define UPNP_E_FINISH -116
00192
00198 #define UPNP_E_INIT_FAILED -117
00199
00205 #define UPNP_E_URL_TOO_BIG -118
00206
00216 #define UPNP_E_BAD_HTTPMSG -119
00217
00224 #define UPNP_E_ALREADY_REGISTERED -120
00225
00230 #define UPNP_E_INVALID_INTERFACE -121
00231
00239 #define UPNP_E_NETWORK_ERROR -200
00240
00249 #define UPNP_E_SOCKET_WRITE -201
00250
00259 #define UPNP_E_SOCKET_READ -202
00260
00269 #define UPNP_E_SOCKET_BIND -203
00270
00279 #define UPNP_E_SOCKET_CONNECT -204
00280
00289 #define UPNP_E_OUTOF_SOCKET -205
00290
00297 #define UPNP_E_LISTEN -206
00298
00305 #define UPNP_E_TIMEDOUT -207
00306
00313 #define UPNP_E_SOCKET_ERROR -208
00314
00315 #define UPNP_E_FILE_WRITE_ERROR -209
00316
00322 #define UPNP_E_CANCELED -210
00323
00324 #define UPNP_E_EVENT_PROTOCOL -300
00325
00329 #define UPNP_E_SUBSCRIBE_UNACCEPTED -301
00330
00334 #define UPNP_E_UNSUBSCRIBE_UNACCEPTED -302
00335
00339 #define UPNP_E_NOTIFY_UNACCEPTED -303
00340
00347 #define UPNP_E_INVALID_ARGUMENT -501
00348
00353 #define UPNP_E_FILE_NOT_FOUND -502
00354
00358 #define UPNP_E_FILE_READ_ERROR -503
00359
00364 #define UPNP_E_EXT_NOT_XML -504
00365
00366 #define UPNP_E_NO_WEB_SERVER -505
00367 #define UPNP_E_OUTOF_BOUNDS -506
00368
00373 #define UPNP_E_NOT_FOUND -507
00374
00379 #define UPNP_E_INTERNAL_ERROR -911
00380
00381 /* SOAP-related error codes */
00382 #define UPNP_SOAP_E_INVALID_ACTION 401
00383 #define UPNP_SOAP_E_INVALID_ARGS 402
00384 #define UPNP_SOAP_E_OUT_OF_SYNC 403
00385 #define UPNP_SOAP_E_INVALID_VAR 404
00386 #define UPNP_SOAP_E_ACTION_FAILED 501
00387
00388 /* @} ErrorCodes */
00389
00390 /*
00391  * Opaque data structures. The following includes are data structures that
00392  * must be externally visible. Since version 1.8.0, only an opaque typedef
00393  * is visible from the outside world. Any operation on these data types
00394  * must be done using the appropriate interface functions.
00395  *
00396  * This policy has the great advantage that it is now possible to change
00397  * the internal implementation of these data structures without breaking
00398  * the API.
00399  */
00400 #include "UpnpActionComplete.h"
00401 #include "UpnpActionRequest.h"
00402 #include "UpnpDiscovery.h"
00403 #include "UpnpEvent.h"
00404 #include "UpnpEventSubscribe.h"
00405 #include "UpnpFileInfo.h"
00406 #include "UpnpStateVarComplete.h"
00407 #include "UpnpStateVarRequest.h"
00408 #include "UpnpSubscriptionRequest.h"
```

```
00409
00416 enum UpnpOpenFileMode
00417 {
00418         UPNP_READ,
00419         UPNP_WRITE
00420 };
00421
00429 typedef int UpnpClient_Handle;
00430
00439 typedef int UpnpDevice_Handle;
00440
00448 typedef char Upnp_SID[44];
00449
00458 enum Upnp_SType_e
00459 {
00461         UPNP_S_ALL,
00462
00464         UPNP_S_ROOT,
00465
00468         UPNP_S_DEVICE,
00469
00472         UPNP_S_SERVICE
00473 };
00474
00475 typedef enum Upnp_SType_e Upnp_SType;
00476
00483 enum Upnp_DescType_e
00484 {
00486         UPNPREG_URL_DESC,
00487
00490         UPNPREG_FILENAME_DESC,
00491
00494         UPNPREG_BUF_DESC
00495 };
00496
00497 typedef enum Upnp_DescType_e Upnp_DescType;
00498
00499 #include "Callback.h"
00500
00501 /* @} Constants and Types */
00502
00503 #ifdef __cplusplus
00504 extern "C" {
00505 #endif /* __cplusplus */
00506
00545 UPNP_EXPORT_SPEC int UpnpInit2(
00549         const char *IfName,
00552         unsigned short DestPort);
00553
00566 #ifdef UPNP_ENABLE_OPEN_SSL
00567 UPNP_EXPORT_SPEC int UpnpInitSslContext(
00572         int initOpenSslLib,
00575         const SSL_METHOD *sslMethod);
00576 #endif
00577
00597 UPNP_EXPORT_SPEC int UpnpFinish(void);
00598
00610 UPNP_EXPORT_SPEC unsigned short UpnpGetServerPort(void);
00611
00623 UPNP_EXPORT_SPEC unsigned short UpnpGetServerPort6(void);
00624
00636 UPNP_EXPORT_SPEC unsigned short UpnpGetServerUlaGuaPort6(void);
00637
00649 UPNP_EXPORT_SPEC char *UpnpGetServerIpAddress(void);
00650
00662 UPNP_EXPORT_SPEC char *UpnpGetServerIp6Address(void);
00663
00676 UPNP_EXPORT_SPEC char *UpnpGetServerUlaGuaIp6Address(void);
00713 UPNP_EXPORT_SPEC int UpnpRegisterRootDevice(
00716         const char *DescUrl,
00719         Upnp_FunPtr Callback,
00722         const void *Cookie,
00724         UpnpDevice_Handle *Hnd);
00725
00794 UPNP_EXPORT_SPEC int UpnpRegisterRootDevice2(
00796         Upnp_DescType descriptionType,
00799         const char *description,
00802         size_t bufferLen,
00805         int config_baseURL,
00808         Upnp_FunPtr Fun,
00811         const void *Cookie,
00813         UpnpDevice_Handle *Hnd);
00814
00850 UPNP_EXPORT_SPEC int UpnpRegisterRootDevice3(
00853         const char *DescUrl,
00856         Upnp_FunPtr Callback,
00859         const void *Cookie,
```

```
00861         UpnpDevice_Handle *Hnd,
00864         int AddressFamily);
00865
00902 UPNP_EXPORT_SPEC int UpnpRegisterRootDevice4(
00905         const char *DescUrl,
00908         Upnp_FunPtr Callback,
00911         const void *Cookie,
00913         UpnpDevice_Handle *Hnd,
00916         int AddressFamily,
00919         const char *LowerDescUrl);
00920
00937 UPNP_EXPORT_SPEC int UpnpUnRegisterRootDevice(
00939         UpnpDevice_Handle Hnd);
00940
00960 UPNP_EXPORT_SPEC int UpnpUnRegisterRootDeviceLowPower(
00962         UpnpDevice_Handle Hnd,
00964         int PowerState,
00966         int SleepPeriod,
00968         int RegistrationState);
00969
00988 UPNP_EXPORT_SPEC int UpnpRegisterClient(
00990         Upnp_FunPtr Callback,
00993         const void *Cookie,
00996         UpnpClient_Handle *Hnd);
00997
01014 UPNP_EXPORT_SPEC int UpnpUnRegisterClient(
01016         UpnpClient_Handle Hnd);
01017
01024 UPNP_EXPORT_SPEC int UpnpSetContentLength(
01027         UpnpClient_Handle Hnd,
01029         size_t contentLength);
01030
01047 UPNP_EXPORT_SPEC int UpnpSetMaxContentLength(
01050         size_t contentLength);
01051
01052 /* @} Initialization and Registration */
01053
01054 /*****************************************************************************
01055  *****************************************************************************
01056  *                                                                           *
01057  *                        D I S C O V E R Y                                  *
01058  *                                                                           *
01059  *****************************************************************************
01060  *****************************************************************************/
01061
01091 UPNP_EXPORT_SPEC int UpnpSearchAsync(
01093         UpnpClient_Handle Hnd,
01098         int Mx,
01101         const char *TTarget_constarget_const,
01103         const void *Cookie_const);
01104
01120 UPNP_EXPORT_SPEC int UpnpSendAdvertisement(
01122         UpnpDevice_Handle Hnd,
01128         int Exp);
01129
01148 UPNP_EXPORT_SPEC int UpnpSendAdvertisementLowPower(
01150         UpnpDevice_Handle Hnd,
01156         int Exp,
01158         int PowerState,
01160         int SleepPeriod,
01162         int RegistrationState);
01163
01164 /* @} Discovery */
01165
01166 /*****************************************************************************
01167  *****************************************************************************
01168  *                                                                           *
01169  *                            C O N T R O L                                  *
01170  *                                                                           *
01171  *****************************************************************************
01172  *****************************************************************************/
01173
01205 UPNP_EXPORT_SPEC int UpnpGetServiceVarStatus(
01207         UpnpClient_Handle Hnd,
01209         const char *ActionURL,
01211         const char *VarName,
01215         DOMString *StVarVal);
01216
01234 UPNP_EXPORT_SPEC int UpnpGetServiceVarStatusAsync(
01236         UpnpClient_Handle Hnd,
01238         const char *ActionURL,
01240         const char *VarName,
01243         Upnp_FunPtr Fun,
01246         const void *Cookie);
01247
01271 UPNP_EXPORT_SPEC int UpnpSendAction(
01273         UpnpClient_Handle Hnd,
```

```
01275         const char *ActionURL,
01277         const char *ServiceType,
01279         const char *DevUDN,
01281         IXML_Document *Action,
01284         IXML_Document **RespNode);
01285
01309 UPNP_EXPORT_SPEC int UpnpSendActionEx(
01311         UpnpClient_Handle Hnd,
01313         const char *ActionURL,
01315         const char *ServiceType,
01317         const char *DevUDN,
01320         IXML_Document *Header,
01322         IXML_Document *Action,
01325         IXML_Document **RespNode);
01326
01348 UPNP_EXPORT_SPEC int UpnpSendActionAsync(
01350         UpnpClient_Handle Hnd,
01352         const char *ActionURL,
01354         const char *ServiceType,
01356         const char *DevUDN,
01358         IXML_Document *Action,
01361         Upnp_FunPtr Fun,
01364         const void *Cookie);
01365
01387 UPNP_EXPORT_SPEC int UpnpSendActionExAsync(
01389         UpnpClient_Handle Hnd,
01391         const char *ActionURL,
01393         const char *ServiceType,
01395         const char *DevUDN,
01398         IXML_Document *Header,
01400         IXML_Document *Action,
01403         Upnp_FunPtr Fun,
01406         const void *Cookie);
01407
01410 /***************************************************************************
01411  ***************************************************************************
01412  *                                                                         *
01413  *                       E V E N T I N G                                   *
01414  *                                                                         *
01415  ***************************************************************************
01416  ***************************************************************************/
01417
01449 UPNP_EXPORT_SPEC int UpnpAcceptSubscription(
01451         UpnpDevice_Handle Hnd,
01454         const char *DevID,
01457         const char *ServID,
01459         const char **VarName,
01461         const char **NewVal,
01463         int cVariables,
01465         const Upnp_SID SubsId);
01466
01489 UPNP_EXPORT_SPEC int UpnpAcceptSubscriptionExt(
01491         UpnpDevice_Handle Hnd,
01494         const char *DevID,
01497         const char *ServID,
01501         IXML_Document *PropSet,
01503         const Upnp_SID SubsId);
01504
01526 UPNP_EXPORT_SPEC int UpnpNotify(
01528         UpnpDevice_Handle,
01531         const char *DevID,
01533         const char *ServID,
01535         const char **VarName,
01537         const char **NewVal,
01539         int cVariables);
01540
01562 UPNP_EXPORT_SPEC int UpnpNotifyExt(
01564         UpnpDevice_Handle,
01567         const char *DevID,
01569         const char *ServID,
01573         IXML_Document *PropSet);
01574
01603 UPNP_EXPORT_SPEC int UpnpRenewSubscription(
01606         UpnpClient_Handle Hnd,
01609         int *TimeOut,
01611         const Upnp_SID SubsId);
01612
01660 UPNP_EXPORT_SPEC int UpnpRenewSubscriptionAsync(
01663         UpnpClient_Handle Hnd,
01666         int TimeOut,
01668         Upnp_SID SubsId,
01671         Upnp_FunPtr Fun,
01674         const void *Cookie);
01675
01689 UPNP_EXPORT_SPEC int UpnpSetMaxSubscriptions(
01692         UpnpDevice_Handle Hnd,
01694         int MaxSubscriptions);
```

```
01695
01710 UPNP_EXPORT_SPEC int UpnpSetMaxSubscriptionTimeOut(
01713         UpnpDevice_Handle Hnd,
01715         int MaxSubscriptionTimeOut);
01716
01746 UPNP_EXPORT_SPEC int UpnpSubscribe(
01748         UpnpClient_Handle Hnd,
01750         const char *PublisherUrl,
01754         int *TimeOut,
01756         Upnp_SID SubsId);
01757
01806 UPNP_EXPORT_SPEC int UpnpSubscribeAsync(
01808         UpnpClient_Handle Hnd,
01810         const char *PublisherUrl,
01813         int TimeOut,
01815         Upnp_FunPtr Fun,
01817         const void *Cookie);
01818
01848 UPNP_EXPORT_SPEC int UpnpUnSubscribe(
01850         UpnpClient_Handle Hnd,
01853         const Upnp_SID SubsId);
01854
01903 UPNP_EXPORT_SPEC int UpnpUnSubscribeAsync(
01905         UpnpClient_Handle Hnd,
01908         Upnp_SID SubsId,
01911         Upnp_FunPtr Fun,
01914         const void *Cookie);
01915
01918 /*****************************************************************************
01919  *****************************************************************************
01920  *                                                                           *
01921  *                        C L I E N T - A P I                                *
01922  *                                                                           *
01923  *****************************************************************************
01924  *****************************************************************************/
01925
01935 enum Upnp_HttpMethod_e
01936 {
01937         UPNP_HTTPMETHOD_PUT = 0,
01938         UPNP_HTTPMETHOD_DELETE = 1,
01939         UPNP_HTTPMETHOD_GET = 2,
01940         UPNP_HTTPMETHOD_HEAD = 3,
01941         UPNP_HTTPMETHOD_POST = 4
01942 };
01943
01944 typedef enum Upnp_HttpMethod_e Upnp_HttpMethod;
01945
01972 UPNP_EXPORT_SPEC int UpnpDownloadUrlItem(
01974         const char *url,
01976         char **outBuf,
01979         char *contentType);
01980
02011 UPNP_EXPORT_SPEC int UpnpOpenHttpGet(
02013         const char *url,
02015         void **handle,
02017         char **contentType,
02019         int *contentLength,
02021         int *httpStatus,
02026         int timeout);
02027
02058 UPNP_EXPORT_SPEC int UpnpOpenHttpGetProxy(
02060         const char *url,
02062         const char *proxy_str,
02064         void **handle,
02066         char **contentType,
02068         int *contentLength,
02070         int *httpStatus,
02075         int timeout);
02076
02107 UPNP_EXPORT_SPEC int UpnpOpenHttpGetEx(
02109         const char *url,
02111         void **handle,
02113         char **contentType,
02115         int *contentLength,
02118         int *httpStatus,
02121         int lowRange,
02124         int highRange,
02128         int timeout);
02129
02147 UPNP_EXPORT_SPEC int UpnpReadHttpGet(
02149         void *handle,
02151         char *buf,
02153         size_t *size,
02158         int timeout);
02159
02168 UPNP_EXPORT_SPEC int UpnpHttpGetProgress(
02170         void *handle,
```

```
02172         size_t *length,
02174         size_t *total);
02175
02183 UPNP_EXPORT_SPEC int UpnpCancelHttpGet(
02186         void *handle);
02187
02196 UPNP_EXPORT_SPEC int UpnpCloseHttpGet(
02199         void *handle);
02200
02226 UPNP_EXPORT_SPEC int UpnpOpenHttpPost(
02228         const char *url,
02232         void **handle,
02235         const char *contentType,
02237         int contentLength,
02241         int timeout);
02242
02256 UPNP_EXPORT_SPEC int UpnpWriteHttpPost(
02259         void *handle,
02261         char *buf,
02263         size_t *size,
02267         int timeout);
02268
02282 UPNP_EXPORT_SPEC int UpnpCloseHttpPost(
02285         void *handle,
02288         int *httpStatus,
02292         int timeout);
02293
02317 UPNP_EXPORT_SPEC int UpnpOpenHttpConnection(
02320         const char *url,
02324         void **handle,
02328         int timeout);
02329
02355 UPNP_EXPORT_SPEC int UpnpMakeHttpRequest(
02356         /* ![in] The method to use to make the request. */
02357         Upnp_HttpMethod method,
02361         const char *url,
02363         void *handle,
02368         UpnpString *headers,
02370         const char *contentType,
02375         int contentLength,
02379         int timeout);
02380
02395 UPNP_EXPORT_SPEC int UpnpWriteHttpRequest(
02398         void *handle,
02400         char *buf,
02402         size_t *size,
02406         int timeout);
02407
02426 UPNP_EXPORT_SPEC int UpnpEndHttpRequest(
02428         void *handle,
02432         int timeout);
02433
02462 UPNP_EXPORT_SPEC int UpnpGetHttpResponse(
02465         void *handle,
02468         UpnpString *headers,
02470         char **contentType,
02472         int *contentLength,
02474         int *httpStatus,
02479         int timeout);
02480
02499 UPNP_EXPORT_SPEC int UpnpReadHttpResponse(
02502         void *handle,
02504         char *buf,
02506         size_t *size,
02511         int timeout);
02512
02525 UPNP_EXPORT_SPEC int UpnpCloseHttpConnection(
02528         void *handle);
02529
02557 UPNP_EXPORT_SPEC int UpnpDownloadXmlDoc(
02559         const char *url,
02561         IXML_Document **xmlDoc);
02562
02565 /*****************************************************************************
02566  *****************************************************************************
02567  *                                                                           *
02568  *                   W E B  S E R V E R  A P I                               *
02569  *                                                                           *
02570  *****************************************************************************
02571  *****************************************************************************/
02572
02595 UPNP_EXPORT_SPEC int UpnpSetWebServerRootDir(
02597         const char *rootDir);
02598
02602 typedef void *UpnpWebFileHandle;
02603
02607 typedef int (*VDCallback_GetInfo)(
```

```
02609         const char *filename,
02612         UpnpFileInfo *info,
02614         const void *cookie,
02616         const void **request_cookie);
02617
02626 UPNP_EXPORT_SPEC int UpnpVirtualDir_set_GetInfoCallback(
02627         VDCallback_GetInfo callback);
02628
02632 typedef UpnpWebFileHandle (*VDCallback_Open)(
02634         const char *filename,
02637         enum UpnpOpenFileMode Mode,
02639         const void *cookie,
02641         const void *request_cookie);
02642
02651 UPNP_EXPORT_SPEC int UpnpVirtualDir_set_OpenCallback(VDCallback_Open callback);
02652
02656 typedef int (*VDCallback_Read)(
02658         UpnpWebFileHandle fileHnd,
02660         char *buf,
02662         size_t buflen,
02664         const void *cookie,
02666         const void *request_cookie);
02667
02676 UPNP_EXPORT_SPEC int UpnpVirtualDir_set_ReadCallback(VDCallback_Read callback);
02677
02681 typedef int (*VDCallback_Write)(
02683         UpnpWebFileHandle fileHnd,
02685         char *buf,
02687         size_t buflen,
02689         const void *cookie,
02691         const void *request_cookie);
02692
02701 UPNP_EXPORT_SPEC int UpnpVirtualDir_set_WriteCallback(
02702         VDCallback_Write callback);
02703
02707 typedef int (*VDCallback_Seek)(
02709         UpnpWebFileHandle fileHnd,
02713         off_t offset,
02718         int origin,
02720         const void *cookie,
02722         const void *request_cookie);
02723
02732 UPNP_EXPORT_SPEC int UpnpVirtualDir_set_SeekCallback(VDCallback_Seek callback);
02733
02737 typedef int (*VDCallback_Close)(
02739         UpnpWebFileHandle fileHnd,
02741         const void *cookie,
02743         const void *request_cookie);
02744
02753 UPNP_EXPORT_SPEC int UpnpVirtualDir_set_CloseCallback(
02754         VDCallback_Close callback);
02755
02766 UPNP_EXPORT_SPEC int UpnpEnableWebserver(
02768         int enable);
02769
02777 UPNP_EXPORT_SPEC int UpnpIsWebserverEnabled(void);
02778
02779 /*
02780  * \brief Callback for validating HTTP requests HOST header values.
02781  *
02782  * @param hostname the value in the request HOST header.
02783  * @return An integer representing one of the following:
02784  *     \li \c UPNP_E_SUCCESS: a request with the HOST header set to hostname
02785  *                            should be processed.
02786  *     \li \c UPNP_E_BAD_HTTPMSG the request should be rejected.
02787  */
02788 typedef int (*WebCallback_HostValidate)(const char *hostname, void *cookie);
02789
02790 /*
02791  * \brief Set callback for validating HTTP requests HOST header values.
02792  *
02793  * @param callback the host validating callback function or NULL.
02794  * @param cookie the chocolate you like.
02795  */
02796 UPNP_EXPORT_SPEC void UpnpSetHostValidateCallback(
02797         WebCallback_HostValidate callback, void *cookie);
02798
02799 /*
02800  * \brief Enable or disable literal IP redirection.
02801  *
02802  * @param enable Zero to disable (default) non-zero to enable.
02803  */
02804 UPNP_EXPORT_SPEC void UpnpSetAllowLiteralHostRedirection(int enable);
02805
02817 UPNP_EXPORT_SPEC int UpnpSetWebServerCorsString(
02819         const char *corsString);
02820
```

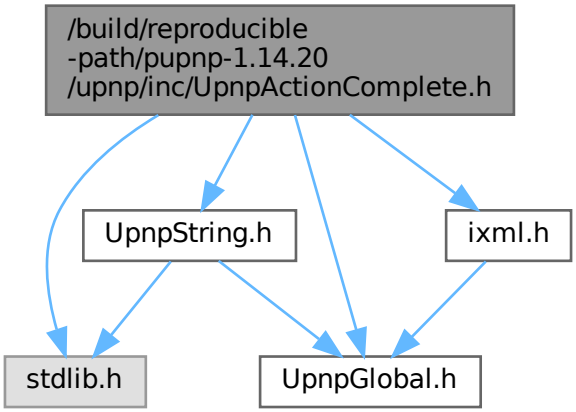
```
02835 UPNP_EXPORT_SPEC int UpnpAddVirtualDir(
02837         const char *dirName,
02839         const void *cookie,
02842         const void **oldcookie);
02843
02851 UPNP_EXPORT_SPEC int UpnpRemoveVirtualDir(
02853         const char *dirName);
02854
02858 UPNP_EXPORT_SPEC void UpnpRemoveAllVirtualDirs(void);
02859
02860 /* @} Web Server API */
02861
02862 #ifdef __cplusplus
02863 }
02864 #endif /* __cplusplus */
02865
02866 /* @} UPnPAPI UPnP API */
02867
02868 #endif /* UPNP_H */
```

9.31 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionComplete.h File Reference

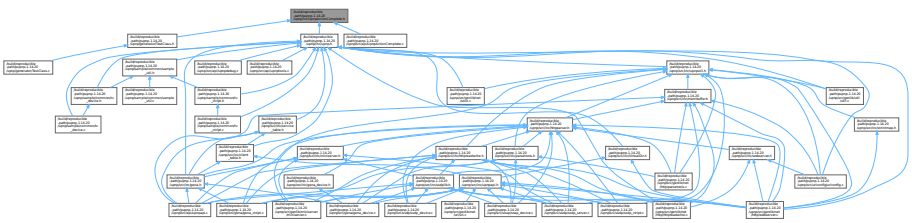
Header file for UpnpActionComplete methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpString.h"
#include "ixml.h"
```

Include dependency graph for UpnpActionComplete.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpActionComplete UpnpActionComplete

Functions

- UpnpActionComplete * UpnpActionComplete_new ()
- void UpnpActionComplete_delete (UpnpActionComplete *p)
- UpnpActionComplete * UpnpActionComplete_dup (const UpnpActionComplete *p)
- int UpnpActionComplete_assign (UpnpActionComplete *p, const UpnpActionComplete *q)
- int UpnpActionComplete_get_ErrCode (const UpnpActionComplete *p)
- int UpnpActionComplete_set_ErrCode (UpnpActionComplete *p, int n)
- const UpnpString * UpnpActionComplete_get_CtrlUrl (const UpnpActionComplete *p)
- int UpnpActionComplete_set_CtrlUrl (UpnpActionComplete *p, const UpnpString *s)
- size_t UpnpActionComplete_get_CtrlUrl_Length (const UpnpActionComplete *p)
- const char * UpnpActionComplete_get_CtrlUrl_cstr (const UpnpActionComplete *p)
- int UpnpActionComplete_strcpy_CtrlUrl (UpnpActionComplete *p, const char *s)
- int UpnpActionComplete_strncpy_CtrlUrl (UpnpActionComplete *p, const char *s, size_t n)
- void UpnpActionComplete_clear_CtrlUrl (UpnpActionComplete *p)
- IXML_Document * UpnpActionComplete_get_ActionRequest (const UpnpActionComplete *p)
- int UpnpActionComplete_set_ActionRequest (UpnpActionComplete *p, IXML_Document *n)
- IXML_Document * UpnpActionComplete_get_ActionResult (const UpnpActionComplete *p)
- int UpnpActionComplete_set_ActionResult (UpnpActionComplete *p, IXML_Document *n)

9.31.1 Detailed Description

Header file for UpnpActionComplete methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.31.2 Typedef Documentation

9.31.2.1 UpnpActionComplete

```
typedef struct s_UpnpActionComplete UpnpActionComplete
```

UpnpActionComplete

9.31.3 Function Documentation

9.31.3.1 UpnpActionComplete_assign()

```
int UpnpActionComplete_assign (
            UpnpActionComplete * p,
            const UpnpActionComplete * q )
```

Assignment operator

References UpnpActionComplete_get_ActionRequest(), UpnpActionComplete_get_ActionResult(), UpnpActionComplete_get_CtrlUrl(), UpnpActionComplete_get_ErrCode(), UpnpActionComplete_set_ActionRequest(), UpnpActionComplete_set_ActionResult(), UpnpActionComplete_set_CtrlUrl(), and UpnpActionComplete_set_ErrCode().

Referenced by UpnpActionComplete_dup().

9.31.3.2 UpnpActionComplete_clear_CtrlUrl()

```
void UpnpActionComplete_clear_CtrlUrl (
            UpnpActionComplete * p )
```

UpnpActionComplete_clear_CtrlUrl

References UpnpString_clear().

9.31.3.3 UpnpActionComplete_delete()

```
void UpnpActionComplete_delete (
            UpnpActionComplete * p )
```

Destructor

References UpnpString_delete().

Referenced by TestClass_delete(), TestClass_set_TheActionComplete(), and UpnpThreadDistribution().

9.31.3.4 UpnpActionComplete_dup()

```
UpnpActionComplete * UpnpActionComplete_dup (
            const UpnpActionComplete * p )
```

Copy Constructor

References UpnpActionComplete_assign(), and UpnpActionComplete_new().

Referenced by TestClass_set_TheActionComplete().

9.31.3.5 UpnpActionComplete_get_ActionRequest()

```
IXML_Document * UpnpActionComplete_get_ActionRequest (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_ActionRequest

Referenced by SampleUtil_PrintEvent(), and UpnpActionComplete_assign().

9.31.3.6 UpnpActionComplete_get_ActionResult()

```
IXML_Document * UpnpActionComplete_get_ActionResult (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_ActionResult

Referenced by SampleUtil_PrintEvent(), and UpnpActionComplete_assign().

9.31.3.7 UpnpActionComplete_get_CtrlUrl()

```
const UpnpString * UpnpActionComplete_get_CtrlUrl (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_CtrlUrl

Referenced by SampleUtil_PrintEvent(), UpnpActionComplete_assign(), UpnpActionComplete_get_CtrlUrl_cstr(), and UpnpActionComplete_get_CtrlUrl_Length().

9.31.3.8 UpnpActionComplete_get_CtrlUrl_cstr()

```
const char * UpnpActionComplete_get_CtrlUrl_cstr (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_CtrlUrl_cstr

References UpnpActionComplete_get_CtrlUrl(), and UpnpString_get_String().

9.31.3.9 UpnpActionComplete_get_CtrlUrl_Length()

```
size_t UpnpActionComplete_get_CtrlUrl_Length (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_CtrlUrl_Length

References UpnpActionComplete_get_CtrlUrl(), and UpnpString_get_Length().

9.31.3.10 UpnpActionComplete_get_ErrCode()

```
int UpnpActionComplete_get_ErrCode (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpActionComplete_assign().

9.31.3.11 UpnpActionComplete_new()

```
UpnpActionComplete * UpnpActionComplete_new ( )
```

Constructor

References UpnpString_new().

Referenced by TestClass_new(), UpnpActionComplete_dup(), and UpnpThreadDistribution().

9.31.3.12 UpnpActionComplete_set_ActionRequest()

```
int UpnpActionComplete_set_ActionRequest (
            UpnpActionComplete * p,
            IXML_Document * n )
```

UpnpActionComplete_set_ActionRequest

Referenced by UpnpActionComplete_assign(), and UpnpThreadDistribution().

9.31.3.13 UpnpActionComplete_set_ActionResult()

```
int UpnpActionComplete_set_ActionResult (
            UpnpActionComplete * p,
            IXML_Document * n )
```

UpnpActionComplete_set_ActionResult

Referenced by UpnpActionComplete_assign(), and UpnpThreadDistribution().

9.31.3.14 UpnpActionComplete_set_CtrlUrl()

```
int UpnpActionComplete_set_CtrlUrl (
            UpnpActionComplete * p,
            const UpnpString * s )
```

UpnpActionComplete_set_CtrlUrl

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionComplete_assign().

9.31.3.15 UpnpActionComplete_set_ErrCode()

```
int UpnpActionComplete_set_ErrCode (
            UpnpActionComplete * p,
            int n )
```

UpnpActionComplete_set_ErrCode

Referenced by UpnpActionComplete_assign(), and UpnpThreadDistribution().

9.31.3.16 UpnpActionComplete_strcpy_CtrlUrl()

```
int UpnpActionComplete_strcpy_CtrlUrl (
            UpnpActionComplete * p,
            const char * s )
```

UpnpActionComplete_strcpy_CtrlUrl

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.31.3.17 UpnpActionComplete_strncpy_CtrlUrl()

```
int UpnpActionComplete_strncpy_CtrlUrl (
             UpnpActionComplete * p,
             const char * s,
             size_t n )
```

UpnpActionComplete_strncpy_CtrlUrl

References UpnpString_set_StringN().

9.32 UpnpActionComplete.h

Go to the documentation of this file.

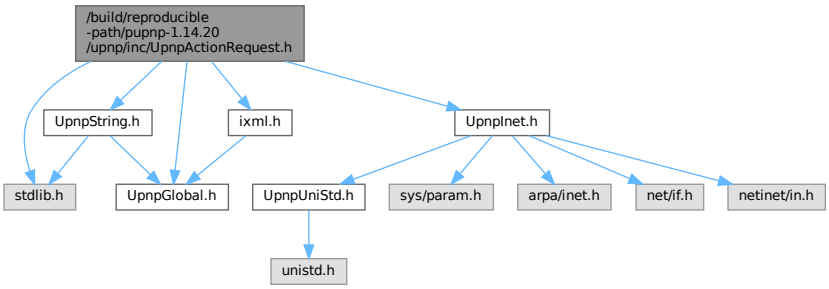
```
00001 #ifndef UPNPACTIONCOMPLETE_H
00002 #define UPNPACTIONCOMPLETE_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpString.h"
00019 #include "ixml.h"
00020
00021 #ifdef __cplusplus
00022 extern "C" {
00023 #endif /* __cplusplus */
00024
00028 typedef struct s_UpnpActionComplete UpnpActionComplete;
00029
00031 UPNP_EXPORT_SPEC UpnpActionComplete *UpnpActionComplete_new();
00033 UPNP_EXPORT_SPEC void UpnpActionComplete_delete(UpnpActionComplete *p);
00035 UPNP_EXPORT_SPEC UpnpActionComplete *UpnpActionComplete_dup(
00036         const UpnpActionComplete *p);
00038 UPNP_EXPORT_SPEC int UpnpActionComplete_assign(
00039         UpnpActionComplete *p, const UpnpActionComplete *q);
00040
00042 UPNP_EXPORT_SPEC int UpnpActionComplete_get_ErrCode(
00043         const UpnpActionComplete *p);
00045 UPNP_EXPORT_SPEC int UpnpActionComplete_set_ErrCode(
00046         UpnpActionComplete *p, int n);
00047
00049 UPNP_EXPORT_SPEC const UpnpString *UpnpActionComplete_get_CtrlUrl(
00050         const UpnpActionComplete *p);
00052 UPNP_EXPORT_SPEC int UpnpActionComplete_set_CtrlUrl(
00053         UpnpActionComplete *p, const UpnpString *s);
00055 UPNP_EXPORT_SPEC size_t UpnpActionComplete_get_CtrlUrl_Length(
00056         const UpnpActionComplete *p);
00058 UPNP_EXPORT_SPEC const char *UpnpActionComplete_get_CtrlUrl_cstr(
00059         const UpnpActionComplete *p);
00061 UPNP_EXPORT_SPEC int UpnpActionComplete_strcpy_CtrlUrl(
00062         UpnpActionComplete *p, const char *s);
00064 UPNP_EXPORT_SPEC int UpnpActionComplete_strncpy_CtrlUrl(
00065         UpnpActionComplete *p, const char *s, size_t n);
00067 UPNP_EXPORT_SPEC void UpnpActionComplete_clear_CtrlUrl(UpnpActionComplete *p);
00068
00070 UPNP_EXPORT_SPEC IXML_Document *UpnpActionComplete_get_ActionRequest(
00071         const UpnpActionComplete *p);
00073 UPNP_EXPORT_SPEC int UpnpActionComplete_set_ActionRequest(
00074         UpnpActionComplete *p, IXML_Document *n);
00075
00077 UPNP_EXPORT_SPEC IXML_Document *UpnpActionComplete_get_ActionResult(
00078         const UpnpActionComplete *p);
00080 UPNP_EXPORT_SPEC int UpnpActionComplete_set_ActionResult(
00081         UpnpActionComplete *p, IXML_Document *n);
00082
00083 #ifdef __cplusplus
00084 }
00085 #endif /* __cplusplus */
00086
00087 #endif /* UPNPACTIONCOMPLETE_H */
```

9.33 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionRequest.h File Reference

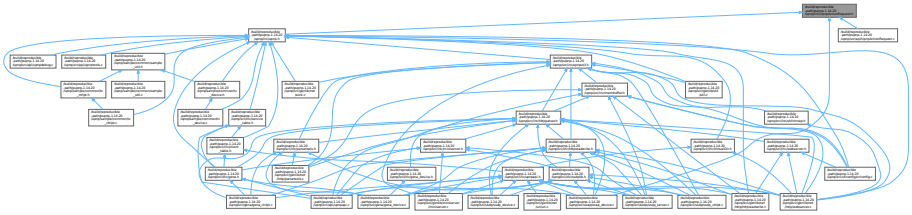
Header file for UpnpActionRequest methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include "UpnpString.h"
#include "ixml.h"
```

Include dependency graph for UpnpActionRequest.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpActionRequest UpnpActionRequest

Functions

- UpnpActionRequest * UpnpActionRequest_new ()
- void UpnpActionRequest_delete (UpnpActionRequest *p)
- UpnpActionRequest * UpnpActionRequest_dup (const UpnpActionRequest *p)
- int UpnpActionRequest_assign (UpnpActionRequest *p, const UpnpActionRequest *q)
- int UpnpActionRequest_get_ErrCode (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ErrCode (UpnpActionRequest *p, int n)
- int UpnpActionRequest_get_Socket (const UpnpActionRequest *p)
- int UpnpActionRequest_set_Socket (UpnpActionRequest *p, int n)
- const UpnpString * UpnpActionRequest_get_ErrStr (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ErrStr (UpnpActionRequest *p, const UpnpString *s)

- size_t UpnpActionRequest_get_ErrStr_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_ErrStr_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_ErrStr (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_ErrStr (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_ErrStr (UpnpActionRequest *p)
- const UpnpString * UpnpActionRequest_get_ActionName (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ActionName (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_ActionName_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_ActionName_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_ActionName (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_ActionName (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_ActionName (UpnpActionRequest *p)
- const UpnpString * UpnpActionRequest_get_DevUDN (const UpnpActionRequest *p)
- int UpnpActionRequest_set_DevUDN (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_DevUDN_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_DevUDN_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_DevUDN (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_DevUDN (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_DevUDN (UpnpActionRequest *p)
- const UpnpString * UpnpActionRequest_get_ServiceID (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ServiceID (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_ServiceID_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_ServiceID_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_ServiceID (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_ServiceID (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_ServiceID (UpnpActionRequest *p)
- IXML_Document * UpnpActionRequest_get_ActionRequest (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ActionRequest (UpnpActionRequest *p, IXML_Document *n)
- IXML_Document * UpnpActionRequest_get_ActionResult (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ActionResult (UpnpActionRequest *p, IXML_Document *n)
- IXML_Document * UpnpActionRequest_get_SoapHeader (const UpnpActionRequest *p)
- int UpnpActionRequest_set_SoapHeader (UpnpActionRequest *p, IXML_Document *n)
- const struct sockaddr_storage * UpnpActionRequest_get_CtrlPtIPAddr (const UpnpActionRequest *p)
- int UpnpActionRequest_set_CtrlPtIPAddr (UpnpActionRequest *p, const struct sockaddr_storage *buf)
- void UpnpActionRequest_clear_CtrlPtIPAddr (UpnpActionRequest *p)
- const UpnpString * UpnpActionRequest_get_Os (const UpnpActionRequest *p)
- int UpnpActionRequest_set_Os (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_Os_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_Os_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_Os (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_Os (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_Os (UpnpActionRequest *p)

9.33.1 Detailed Description

Header file for UpnpActionRequest methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.33.2 Typedef Documentation

9.33.2.1 UpnpActionRequest

```
typedef struct s_UpnpActionRequest UpnpActionRequest
```

UpnpActionRequest

9.33.3 Function Documentation

9.33.3.1 UpnpActionRequest_assign()

```
int UpnpActionRequest_assign (
            UpnpActionRequest * p,
            const UpnpActionRequest * q )
```

Assignment operator

References UpnpActionRequest_get_ActionName(), UpnpActionRequest_get_ActionRequest(), UpnpActionRequest_get_ActionResu
UpnpActionRequest_get_CtrlPtIPAddr(), UpnpActionRequest_get_DevUDN(), UpnpActionRequest_get_ErrCode(),
UpnpActionRequest_get_ErrStr(), UpnpActionRequest_get_Os(), UpnpActionRequest_get_ServiceID(), UpnpActionRequest_get_So
UpnpActionRequest_get_Socket(), UpnpActionRequest_set_ActionName(), UpnpActionRequest_set_ActionRequest(),
UpnpActionRequest_set_ActionResult(), UpnpActionRequest_set_CtrlPtIPAddr(), UpnpActionRequest_set_DevUDN(),
UpnpActionRequest_set_ErrCode(), UpnpActionRequest_set_ErrStr(), UpnpActionRequest_set_Os(), UpnpActionRequest_set_Serv
UpnpActionRequest_set_SoapHeader(), and UpnpActionRequest_set_Socket().

Referenced by UpnpActionRequest_dup().

9.33.3.2 UpnpActionRequest_clear_ActionName()

```
void UpnpActionRequest_clear_ActionName (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_ActionName

References UpnpString_clear().

9.33.3.3 UpnpActionRequest_clear_CtrlPtIPAddr()

```
void UpnpActionRequest_clear_CtrlPtIPAddr (
            UpnpActionRequest * p )
```

UpnpActionRequest_get_CtrlPtIPAddr

9.33.3.4 UpnpActionRequest_clear_DevUDN()

```
void UpnpActionRequest_clear_DevUDN (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_DevUDN

References UpnpString_clear().

9.33.3.5 UpnpActionRequest_clear_ErrStr()

```
void UpnpActionRequest_clear_ErrStr (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_ErrStr

References UpnpString_clear().

9.33.3.6 UpnpActionRequest_clear_Os()

```
void UpnpActionRequest_clear_Os (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_Os

References UpnpString_clear().

9.33.3.7 UpnpActionRequest_clear_ServiceID()

```
void UpnpActionRequest_clear_ServiceID (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_ServiceID

References UpnpString_clear().

9.33.3.8 UpnpActionRequest_delete()

```
void UpnpActionRequest_delete (
            UpnpActionRequest * p )
```

Destructor

References UpnpString_delete().

Referenced by handle_invoke_action().

9.33.3.9 UpnpActionRequest_dup()

```
UpnpActionRequest * UpnpActionRequest_dup (
            const UpnpActionRequest * p )
```

Copy Constructor

References UpnpActionRequest_assign(), and UpnpActionRequest_new().

9.33.3.10 UpnpActionRequest_get_ActionName()

```
const UpnpString * UpnpActionRequest_get_ActionName (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionName

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), UpnpActionRequest_assign(), UpnpActionRequest_get_ActionName_cstr(), and UpnpActionRequest_get_ActionName_Length().

9.33.3.11 UpnpActionRequest_get_ActionName_cstr()

```
const char * UpnpActionRequest_get_ActionName_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionName_cstr

References UpnpActionRequest_get_ActionName(), and UpnpString_get_String().

9.33.3.12 UpnpActionRequest_get_ActionName_Length()

```
size_t UpnpActionRequest_get_ActionName_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionName_Length

References UpnpActionRequest_get_ActionName(), and UpnpString_get_Length().

9.33.3.13 UpnpActionRequest_get_ActionRequest()

```
IXML_Document * UpnpActionRequest_get_ActionRequest (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionRequest

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.33.3.14 UpnpActionRequest_get_ActionResult()

```
IXML_Document * UpnpActionRequest_get_ActionResult (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionResult

Referenced by handle_invoke_action(), SampleUtil_PrintEvent(), and UpnpActionRequest_assign().

9.33.3.15 UpnpActionRequest_get_CtrlPtIPAddr()

```
const struct sockaddr_storage * UpnpActionRequest_get_CtrlPtIPAddr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_CtrlPtIPAddr

Referenced by UpnpActionRequest_assign().

9.33.3.16 UpnpActionRequest_get_DevUDN()

```
const UpnpString * UpnpActionRequest_get_DevUDN (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_DevUDN

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), UpnpActionRequest_assign(), UpnpActionRequest_get_DevUDN_cstr(), and UpnpActionRequest_get_DevUDN_Length().

9.33.3.17 UpnpActionRequest_get_DevUDN_cstr()

```
const char * UpnpActionRequest_get_DevUDN_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_DevUDN_cstr

References UpnpActionRequest_get_DevUDN(), and UpnpString_get_String().

9.33.3.18 UpnpActionRequest_get_DevUDN_Length()

```
size_t UpnpActionRequest_get_DevUDN_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_DevUDN_Length

References UpnpActionRequest_get_DevUDN(), and UpnpString_get_Length().

9.33.3.19 UpnpActionRequest_get_ErrCode()

```
int UpnpActionRequest_get_ErrCode (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrCode

Referenced by handle_invoke_action(), SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.33.3.20 UpnpActionRequest_get_ErrStr()

```
const UpnpString * UpnpActionRequest_get_ErrStr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrStr

Referenced by SampleUtil_PrintEvent(), UpnpActionRequest_assign(), UpnpActionRequest_get_ErrStr_cstr(), and UpnpActionRequest_get_ErrStr_Length().

9.33.3.21 UpnpActionRequest_get_ErrStr_cstr()

```
const char * UpnpActionRequest_get_ErrStr_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrStr_cstr

References UpnpActionRequest_get_ErrStr(), and UpnpString_get_String().

Referenced by handle_invoke_action().

9.33.3.22 UpnpActionRequest_get_ErrStr_Length()

```
size_t UpnpActionRequest_get_ErrStr_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrStr_Length

References UpnpActionRequest_get_ErrStr(), and UpnpString_get_Length().

9.33.3.23 UpnpActionRequest_get_Os()

```
const UpnpString * UpnpActionRequest_get_Os (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Os

Referenced by UpnpActionRequest_assign(), UpnpActionRequest_get_Os_cstr(), and UpnpActionRequest_get_Os_Length().

9.33.3.24 UpnpActionRequest_get_Os_cstr()

```
const char * UpnpActionRequest_get_Os_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Os_cstr

References UpnpActionRequest_get_Os(), and UpnpString_get_String().

9.33.3.25 UpnpActionRequest_get_Os_Length()

```
size_t UpnpActionRequest_get_Os_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Os_Length

References UpnpActionRequest_get_Os(), and UpnpString_get_Length().

9.33.3.26 UpnpActionRequest_get_ServiceID()

```
const UpnpString * UpnpActionRequest_get_ServiceID (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ServiceID

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), UpnpActionRequest_assign(), UpnpActionRequest_get_ServiceID_cstr(), and UpnpActionRequest_get_ServiceID_Length().

9.33.3.27 UpnpActionRequest_get_ServiceID_cstr()

```
const char * UpnpActionRequest_get_ServiceID_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ServiceID_cstr

References UpnpActionRequest_get_ServiceID(), and UpnpString_get_String().

9.33.3.28 UpnpActionRequest_get_ServiceID_Length()

```
size_t UpnpActionRequest_get_ServiceID_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ServiceID_Length

References UpnpActionRequest_get_ServiceID(), and UpnpString_get_Length().

9.33.3.29 UpnpActionRequest_get_SoapHeader()

```
IXML_Document * UpnpActionRequest_get_SoapHeader (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_SoapHeader

Referenced by UpnpActionRequest_assign().

9.33.3.30 UpnpActionRequest_get_Socket()

```
int UpnpActionRequest_get_Socket (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Socket

Referenced by UpnpActionRequest_assign().

9.33.3.31 UpnpActionRequest_new()

```
UpnpActionRequest * UpnpActionRequest_new ( )
```

Constructor

References UpnpString_new().

Referenced by handle_invoke_action(), and UpnpActionRequest_dup().

9.33.3.32 UpnpActionRequest_set_ActionName()

```
int UpnpActionRequest_set_ActionName (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_ActionName

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.33.3.33 UpnpActionRequest_set_ActionRequest()

```
int UpnpActionRequest_set_ActionRequest (
            UpnpActionRequest * p,
            IXML_Document * n )
```

UpnpActionRequest_set_ActionRequest

Referenced by handle_invoke_action(), and UpnpActionRequest_assign().

9.33.3.34 UpnpActionRequest_set_ActionResult()

```
int UpnpActionRequest_set_ActionResult (
            UpnpActionRequest * p,
            IXML_Document * n )
```

UpnpActionRequest_set_ActionResult

Referenced by handle_invoke_action(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.33.3.35 UpnpActionRequest_set_CtrlPtIPAddr()

```
int UpnpActionRequest_set_CtrlPtIPAddr (
            UpnpActionRequest * p,
            const struct sockaddr_storage * buf )
```

UpnpActionRequest_get_CtrlPtIPAddr

Referenced by handle_invoke_action(), and UpnpActionRequest_assign().

9.33.3.36 UpnpActionRequest_set_DevUDN()

```
int UpnpActionRequest_set_DevUDN (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_DevUDN

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.33.3.37 UpnpActionRequest_set_ErrCode()

```
int UpnpActionRequest_set_ErrCode (
            UpnpActionRequest * p,
            int n )
```

UpnpActionRequest_set_ErrCode

Referenced by handle_invoke_action(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.33.3.38 UpnpActionRequest_set_ErrStr()

```
int UpnpActionRequest_set_ErrStr (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_ErrStr

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.33.3.39 UpnpActionRequest_set_Os()

```
int UpnpActionRequest_set_Os (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_Os

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.33.3.40 UpnpActionRequest_set_ServiceID()

```
int UpnpActionRequest_set_ServiceID (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_ServiceID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.33.3.41 UpnpActionRequest_set_SoapHeader()

```
int UpnpActionRequest_set_SoapHeader (
            UpnpActionRequest * p,
            IXML_Document * n )
```

UpnpActionRequest_set_SoapHeader

Referenced by UpnpActionRequest_assign().

9.33.3.42 UpnpActionRequest_set_Socket()

```
int UpnpActionRequest_set_Socket (
            UpnpActionRequest * p,
            int n )
```

UpnpActionRequest_set_Socket

Referenced by UpnpActionRequest_assign().

9.33.3.43 UpnpActionRequest_strcpy_ActionName()

```
int UpnpActionRequest_strcpy_ActionName (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_ActionName

References UpnpString_set_String().

Referenced by handle_invoke_action().

9.33.3.44 UpnpActionRequest_strcpy_DevUDN()

```
int UpnpActionRequest_strcpy_DevUDN (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_DevUDN

References UpnpString_set_String().

Referenced by handle_invoke_action().

9.33.3.45 UpnpActionRequest_strcpy_ErrStr()

```
int UpnpActionRequest_strcpy_ErrStr (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_ErrStr

References UpnpString_set_String().

Referenced by TvDeviceHandleActionRequest().

9.33.3.46 UpnpActionRequest_strcpy_Os()

```
int UpnpActionRequest_strcpy_Os (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_Os

References UpnpString_set_String().

9.33.3.47 UpnpActionRequest_strcpy_ServiceID()

```
int UpnpActionRequest_strcpy_ServiceID (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_ServiceID

References UpnpString_set_String().

Referenced by handle_invoke_action().

9.33.3.48 UpnpActionRequest_strncpy_ActionName()

```
int UpnpActionRequest_strncpy_ActionName (
            UpnpActionRequest * p,
            const char * s,
            size_t n )
```

UpnpActionRequest_strncpy_ActionName

References UpnpString_set_StringN().

9.33.3.49 UpnpActionRequest_strncpy_DevUDN()

```
int UpnpActionRequest_strncpy_DevUDN (
            UpnpActionRequest * p,
            const char * s,
            size_t n )
```

UpnpActionRequest_strncpy_DevUDN

References UpnpString_set_StringN().

9.33.3.50 UpnpActionRequest_strncpy_ErrStr()

```
int UpnpActionRequest_strncpy_ErrStr (
             UpnpActionRequest * p,
             const char * s,
             size_t n )
```

UpnpActionRequest_strncpy_ErrStr

References UpnpString_set_StringN().

9.33.3.51 UpnpActionRequest_strncpy_Os()

```
int UpnpActionRequest_strncpy_Os (
             UpnpActionRequest * p,
             const char * s,
             size_t n )
```

UpnpActionRequest_strncpy_Os

References UpnpString_set_StringN().

Referenced by handle_invoke_action().

9.33.3.52 UpnpActionRequest_strncpy_ServiceID()

```
int UpnpActionRequest_strncpy_ServiceID (
             UpnpActionRequest * p,
             const char * s,
             size_t n )
```

UpnpActionRequest_strncpy_ServiceID

References UpnpString_set_StringN().

9.34 UpnpActionRequest.h

Go to the documentation of this file.

```
00001 #ifndef UPNPACTIONREQUEST_H
00002 #define UPNPACTIONREQUEST_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpInet.h"
00019 #include "UpnpString.h"
00020 #include "ixml.h"
00021
00022 #ifdef __cplusplus
00023 extern "C" {
00024 #endif /* __cplusplus */
00025
00029 typedef struct s_UpnpActionRequest UpnpActionRequest;
00030
00032 UPNP_EXPORT_SPEC UpnpActionRequest *UpnpActionRequest_new();
00034 UPNP_EXPORT_SPEC void UpnpActionRequest_delete(UpnpActionRequest *p);
00036 UPNP_EXPORT_SPEC UpnpActionRequest *UpnpActionRequest_dup(
00037         const UpnpActionRequest *p);
```

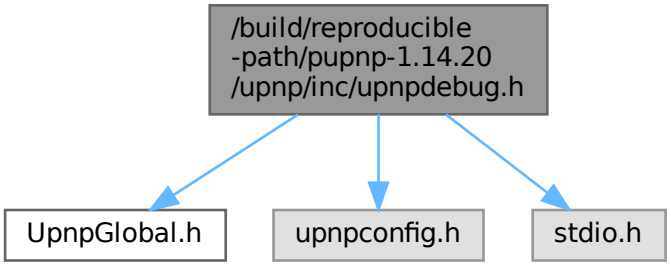
```
00039 UPNP_EXPORT_SPEC int UpnpActionRequest_assign(
00040         UpnpActionRequest *p, const UpnpActionRequest *q);
00041
00043 UPNP_EXPORT_SPEC int UpnpActionRequest_get_ErrCode(const UpnpActionRequest *p);
00045 UPNP_EXPORT_SPEC int UpnpActionRequest_set_ErrCode(UpnpActionRequest *p, int n);
00046
00048 UPNP_EXPORT_SPEC int UpnpActionRequest_get_Socket(const UpnpActionRequest *p);
00050 UPNP_EXPORT_SPEC int UpnpActionRequest_set_Socket(UpnpActionRequest *p, int n);
00051
00053 UPNP_EXPORT_SPEC const UpnpString *UpnpActionRequest_get_ErrStr(
00054         const UpnpActionRequest *p);
00056 UPNP_EXPORT_SPEC int UpnpActionRequest_set_ErrStr(
00057         UpnpActionRequest *p, const UpnpString *s);
00059 UPNP_EXPORT_SPEC size_t UpnpActionRequest_get_ErrStr_Length(
00060         const UpnpActionRequest *p);
00062 UPNP_EXPORT_SPEC const char *UpnpActionRequest_get_ErrStr_cstr(
00063         const UpnpActionRequest *p);
00065 UPNP_EXPORT_SPEC int UpnpActionRequest_strcpy_ErrStr(
00066         UpnpActionRequest *p, const char *s);
00068 UPNP_EXPORT_SPEC int UpnpActionRequest_strncpy_ErrStr(
00069         UpnpActionRequest *p, const char *s, size_t n);
00071 UPNP_EXPORT_SPEC void UpnpActionRequest_clear_ErrStr(UpnpActionRequest *p);
00072
00074 UPNP_EXPORT_SPEC const UpnpString *UpnpActionRequest_get_ActionName(
00075         const UpnpActionRequest *p);
00077 UPNP_EXPORT_SPEC int UpnpActionRequest_set_ActionName(
00078         UpnpActionRequest *p, const UpnpString *s);
00080 UPNP_EXPORT_SPEC size_t UpnpActionRequest_get_ActionName_Length(
00081         const UpnpActionRequest *p);
00083 UPNP_EXPORT_SPEC const char *UpnpActionRequest_get_ActionName_cstr(
00084         const UpnpActionRequest *p);
00086 UPNP_EXPORT_SPEC int UpnpActionRequest_strcpy_ActionName(
00087         UpnpActionRequest *p, const char *s);
00089 UPNP_EXPORT_SPEC int UpnpActionRequest_strncpy_ActionName(
00090         UpnpActionRequest *p, const char *s, size_t n);
00092 UPNP_EXPORT_SPEC void UpnpActionRequest_clear_ActionName(UpnpActionRequest *p);
00093
00095 UPNP_EXPORT_SPEC const UpnpString *UpnpActionRequest_get_DevUDN(
00096         const UpnpActionRequest *p);
00098 UPNP_EXPORT_SPEC int UpnpActionRequest_set_DevUDN(
00099         UpnpActionRequest *p, const UpnpString *s);
00101 UPNP_EXPORT_SPEC size_t UpnpActionRequest_get_DevUDN_Length(
00102         const UpnpActionRequest *p);
00104 UPNP_EXPORT_SPEC const char *UpnpActionRequest_get_DevUDN_cstr(
00105         const UpnpActionRequest *p);
00107 UPNP_EXPORT_SPEC int UpnpActionRequest_strcpy_DevUDN(
00108         UpnpActionRequest *p, const char *s);
00110 UPNP_EXPORT_SPEC int UpnpActionRequest_strncpy_DevUDN(
00111         UpnpActionRequest *p, const char *s, size_t n);
00113 UPNP_EXPORT_SPEC void UpnpActionRequest_clear_DevUDN(UpnpActionRequest *p);
00114
00116 UPNP_EXPORT_SPEC const UpnpString *UpnpActionRequest_get_ServiceID(
00117         const UpnpActionRequest *p);
00119 UPNP_EXPORT_SPEC int UpnpActionRequest_set_ServiceID(
00120         UpnpActionRequest *p, const UpnpString *s);
00122 UPNP_EXPORT_SPEC size_t UpnpActionRequest_get_ServiceID_Length(
00123         const UpnpActionRequest *p);
00125 UPNP_EXPORT_SPEC const char *UpnpActionRequest_get_ServiceID_cstr(
00126         const UpnpActionRequest *p);
00128 UPNP_EXPORT_SPEC int UpnpActionRequest_strcpy_ServiceID(
00129         UpnpActionRequest *p, const char *s);
00131 UPNP_EXPORT_SPEC int UpnpActionRequest_strncpy_ServiceID(
00132         UpnpActionRequest *p, const char *s, size_t n);
00134 UPNP_EXPORT_SPEC void UpnpActionRequest_clear_ServiceID(UpnpActionRequest *p);
00135
00137 UPNP_EXPORT_SPEC IXML_Document *UpnpActionRequest_get_ActionRequest(
00138         const UpnpActionRequest *p);
00140 UPNP_EXPORT_SPEC int UpnpActionRequest_set_ActionRequest(
00141         UpnpActionRequest *p, IXML_Document *n);
00142
00144 UPNP_EXPORT_SPEC IXML_Document *UpnpActionRequest_get_ActionResult(
00145         const UpnpActionRequest *p);
00147 UPNP_EXPORT_SPEC int UpnpActionRequest_set_ActionResult(
00148         UpnpActionRequest *p, IXML_Document *n);
00149
00151 UPNP_EXPORT_SPEC IXML_Document *UpnpActionRequest_get_SoapHeader(
00152         const UpnpActionRequest *p);
00154 UPNP_EXPORT_SPEC int UpnpActionRequest_set_SoapHeader(
00155         UpnpActionRequest *p, IXML_Document *n);
00156
00158 UPNP_EXPORT_SPEC const struct sockaddr_storage *
00159 UpnpActionRequest_get_CtrlPtIPAddr(const UpnpActionRequest *p);
00161 UPNP_EXPORT_SPEC int UpnpActionRequest_set_CtrlPtIPAddr(
00162         UpnpActionRequest *p, const struct sockaddr_storage *buf);
00164 UPNP_EXPORT_SPEC void UpnpActionRequest_clear_CtrlPtIPAddr(
00165         UpnpActionRequest *p);
00166
```

```
00168 UPNP_EXPORT_SPEC const UpnpString *UpnpActionRequest_get_Os(
00169         const UpnpActionRequest *p);
00171 UPNP_EXPORT_SPEC int UpnpActionRequest_set_Os(
00172         UpnpActionRequest *p, const UpnpString *s);
00174 UPNP_EXPORT_SPEC size_t UpnpActionRequest_get_Os_Length(
00175         const UpnpActionRequest *p);
00177 UPNP_EXPORT_SPEC const char *UpnpActionRequest_get_Os_cstr(
00178         const UpnpActionRequest *p);
00180 UPNP_EXPORT_SPEC int UpnpActionRequest_strcpy_Os(
00181         UpnpActionRequest *p, const char *s);
00183 UPNP_EXPORT_SPEC int UpnpActionRequest_strncpy_Os(
00184         UpnpActionRequest *p, const char *s, size_t n);
00186 UPNP_EXPORT_SPEC void UpnpActionRequest_clear_Os(UpnpActionRequest *p);
00187
00188 #ifdef __cplusplus
00189 }
00190 #endif /* __cplusplus */
00191
00192 #endif /* UPNPACTIONREQUEST_H */
```

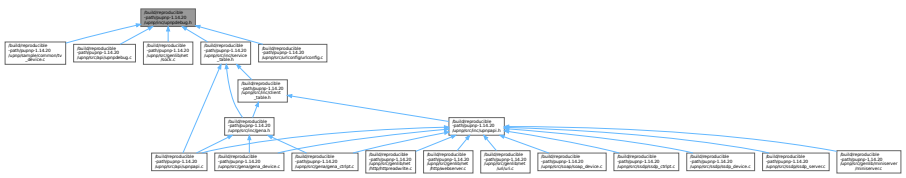
9.35 /build/reproducible-path/pupnp-1.14.20/upnp/inc/upnpdebug.h File Reference

```
#include "UpnpGlobal.h"
#include "upnpconfig.h"
#include <stdio.h>
```

Include dependency graph for upnpdebug.h:



This graph shows which files directly or indirectly include this file:



Upnp_LogLevel

The user has the option to select 4 different types of debugging levels, see `UpnpSetLogLevel`. The critical level will show only those messages which can halt the normal processing of the library, like memory allocation errors.

The remaining three levels are just for debugging purposes. Error will show recoverable errors. Info Level displays the other important operational information regarding the working of the library. If the user selects All, then the library displays all the debugging information that it has.

- UPNP_CRITICAL [0]
- UPNP_ERROR [1]
- UPNP_INFO [2]
- UPNP_ALL [3]

- #define **UPNP_PACKET** UPNP_ERROR
- #define UPNP_DEFAULT_LOG_LEVEL UPNP_ALL
- enum **Upnp_Module** {
  SSDP , **SOAP** , **GENA** , **TPOOL** ,
  MSERV , **DOM** , **API** , **HTTP** }
- enum **Upnp_LogLevel_e** { **UPNP_CRITICAL** , **UPNP_ERROR** , **UPNP_INFO** , **UPNP_ALL** }
- typedef enum Upnp_Module **Dbg_Module**
- typedef enum Upnp_LogLevel_e **Upnp_LogLevel**
- int UpnpInitLog (void)

  Initialize the log files.
- void UpnpSetLogLevel (Upnp_LogLevel log_level)

  Set the log level (see Upnp_LogLevel).
- void UpnpCloseLog (void)

  Closes the log files.
- void UpnpSetLogFileNames (const char *fileName, const char *Ignored)

  Set the name for the log file. There used to be 2 separate files. The second parameter has been kept for compatibility but is ignored. Use a NULL file name for logging to stderr.
- FILE * UpnpGetDebugFile (Upnp_LogLevel level, Dbg_Module module)

  Check if the module is turned on for debug and returns the file descriptor corresponding to the debug level.
- void UpnpPrintf (Upnp_LogLevel DLevel, Dbg_Module Module, const char *DbgFileName, int DbgLineNo, const char *FmtStr,...)

  Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.

9.35.1 Macro Definition Documentation

9.35.1.1 UPNP_DEFAULT_LOG_LEVEL

```
#define UPNP_DEFAULT_LOG_LEVEL UPNP_ALL
```

Default log level : see Upnp_LogLevel

9.35.2 Function Documentation

9.35.2.1 UpnpCloseLog()

```
void UpnpCloseLog (
            void  )
```

Closes the log files.

References GlobalDebugMutex.

Referenced by UpnpFinish().

9.35.2.2 UpnpGetDebugFile()

```
FILE * UpnpGetDebugFile (
            Upnp_LogLevel level,
            Dbg_Module module )
```

Check if the module is turned on for debug and returns the file descriptor corresponding to the debug level.

Returns

NULL if the module is turn off for debug otherwise returns the right FILE pointer.

Parameters

in	*level*	The level of the debug logging. It will decide whether debug statement will go to standard output, or any of the log files.
in	*module*	debug will go in the name of this module.

9.35.2.3 UpnpInitLog()

```
int UpnpInitLog (
            void  )
```

Initialize the log files.

Returns

-1 if fails or UPNP_E_SUCCESS if succeeds.

References GlobalDebugMutex, and UPNP_E_SUCCESS.

Referenced by TvDeviceStart(), and UpnpInitPreamble().

9.35.2.4 UpnpPrintf()

```
void UpnpPrintf (
            Upnp_LogLevel DLevel,
            Dbg_Module Module,
            const char * DbgFileName,
            int DbgLineNo,
            const char * FmtStr,
             ...  )
```

Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.

Parameters

in	*DLevel*	The level of the debug logging. It will decide whether debug statement will go to standard output, or any of the log files.
in	*Module*	debug will go in the name of this module.
in	*DbgFileName*	Name of the file from where debug statement is coming.
in	*DbgLineNo*	Line number of the file from where debug statement is coming.
in	*FmtStr*	Printf like format specification.
in	...	Printf like Variable number of arguments that will go in the debug statement.

References GlobalDebugMutex.

Referenced by AdvertiseAndReply(), AllocGenaHeaders(), create_ssdp_sock_reqv4(), create_ssdp_sock_v4(), DeviceAdvertisement(), DeviceShutdown(), FreeHandle(), gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_validate_delivery_urls(), GenaAutoRenewSubscription(), genaCallback(), genaInitNotify(), genaInitNotifyExt(), genaNotifyAll(), genaNotifyAllExt(), genaRenewSubscription(), genaSubscribe(), genaUnregisterDevice(), get_file_info(), get_miniserver_stopsock(), get_port(), GetHandleInfo(), handle_invoke_action(), handle_query_variable(), http_Connect(), http_MakeMessage(), http_ReadHttpResponse(), http_RecvMessage(), http_RecvPostMessage(), http_SendMessage(), NewRequestHandler(), notify_send_and_recv(), print_http_headers(), PrintHandleInfo(), printService(), printServiceList(), printServiceTable(), PrintThreadPoolStats(), readFromSSDPSocket(), RunMiniServer(), SearchByTarget(), send_action_response(), ServiceAdvertisement(), ServiceShutdown(), sock_destroy(), ssdp_handle_device_request(), start_event_handler(), StopMiniServer(), UpnpAcceptSubscription(), UpnpAcceptSubscriptionExt(), UpnpDownloadXmlDoc(), UpnpFinish(), UpnpGetIfInfo(), UpnpGetServiceVarStatus(), UpnpGetServiceVarStatusAsync(), UpnpInit2(), UpnpInitPreamble(), UpnpInitStartServers(), UpnpNotify(), UpnpNotifyExt(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice3(), UpnpRegisterRootDevice4(), UpnpRenewSubscription(), UpnpRenewSubscriptionAsync(), UpnpSearchAsync(), UpnpSendAction(), UpnpSendActionAsync(), UpnpSendActionEx(), UpnpSendActionExAsync(), UpnpSendAdvertisement(), UpnpSendAdvertisementLowPower(), UpnpSetMaxSubscriptions(), UpnpSetMaxSubscriptionTimeOut(), UpnpSubscribe(), UpnpSubscribeAsync(), UpnpThreadDistribution(), UpnpUnRegisterClient(), UpnpUnRegisterRootDevice(), UpnpUnRegisterRootDeviceLowPower(), UpnpUnSubscribe(), UpnpUnSubscribeAsync(), valid_ssdp_msg(), and web_server_callback().

9.35.2.5 UpnpSetLogFileNames()

```
void UpnpSetLogFileNames (
             const char * fileName,
             const char * Ignored )
```

Set the name for the log file. There used to be 2 separate files. The second parameter has been kept for compatibility but is ignored. Use a NULL file name for logging to stderr.

Parameters

in	*fileName*	Name of the log file.
in	*Ignored*	Ignored.

Referenced by TvDeviceStart().

9.35.2.6 UpnpSetLogLevel()

```
void UpnpSetLogLevel (
             Upnp_LogLevel log_level )
```

Set the log level (see `Upnp_LogLevel`).

Parameters

in	*log_level*	Log level.

References g_log_level.

Referenced by TvDeviceStart().

9.36 upnpdebug.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * Copyright (c) 2006 Rémi Turboult <r3mi@users.sourceforge.net>
00005  * All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * - Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * - Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * - Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef UPNP_DEBUG_H
00034 #define UPNP_DEBUG_H
00035
00040 #include "UpnpGlobal.h" /* for UPNP_INLINE */
00041 #include "upnpconfig.h"
00042
00043 #include <stdio.h>
00044
00045 #ifdef __cplusplus
00046 extern "C" {
00047 #endif
00048
00054
00070 typedef enum Upnp_Module
00071 {
00072         SSDP,
00073         SOAP,
00074         GENA,
00075         TPOOL,
00076         MSERV,
00077         DOM,
00078         API,
00079         HTTP
00080 } Dbg_Module;
00081
00083 typedef enum Upnp_LogLevel_e
00084 {
00085         UPNP_CRITICAL,
00086         UPNP_ERROR,
00087         UPNP_INFO,
00088         UPNP_ALL
00089 } Upnp_LogLevel;
00092 /* UPNP_PACKET probably resulted from a confusion between module and
00093    level and was only used by a few messages in ssdp_device.c (they
00094    have been moved to INFO). Kept for compatibility, don't use for new
00095    messages.
00096 */
00097 #define UPNP_PACKET UPNP_ERROR
00098
00102 #define UPNP_DEFAULT_LOG_LEVEL UPNP_ALL
00103
00109 UPNP_EXPORT_SPEC int UpnpInitLog(void);
00110
00111 #if defined NDEBUG && !defined UPNP_DEBUG_C
00112         #define UpnpInitLog UpnpInitLog_Inlined
00113 static UPNP_INLINE int UpnpInitLog_Inlined(void) { return UPNP_E_SUCCESS; }
00114 #endif
00118 UPNP_EXPORT_SPEC void UpnpSetLogLevel(
00120         Upnp_LogLevel log_level);
00121
```

```
00122 #if defined NDEBUG && !defined UPNP_DEBUG_C
00123         #define UpnpSetLogLevel UpnpSetLogLevel_Inlined
00124 static UPNP_INLINE void UpnpSetLogLevel_Inlined(Upnp_LogLevel log_level)
00125 {
00126         (void)log_level;
00127         return;
00128 }
00129 #endif
00130
00134 void UpnpCloseLog(void);
00135
00136 #if defined NDEBUG && !defined UPNP_DEBUG_C
00137         #define UpnpCloseLog UpnpCloseLog_Inlined
00138 static UPNP_INLINE void UpnpCloseLog_Inlined(void) {}
00139 #endif
00140
00146 UPNP_EXPORT_SPEC void UpnpSetLogFileNames(
00148         const char *fileName,
00150         const char *Ignored);
00151
00152 #if defined NDEBUG && !defined UPNP_DEBUG_C
00153         #define UpnpSetLogFileNames UpnpSetLogFileNames_Inlined
00154 static UPNP_INLINE void UpnpSetLogFileNames_Inlined(
00155         const char *ErrFileName, const char *ignored)
00156 {
00157         (void)ErrFileName;
00158         (void)ignored;
00159         return;
00160 }
00161 #endif
00162
00170 FILE *UpnpGetDebugFile(
00173         Upnp_LogLevel level,
00175         Dbg_Module module);
00176
00177 #if defined NDEBUG && !defined UPNP_DEBUG_C
00178         #define UpnpGetDebugFile UpnpGetDebugFile_Inlined
00179 static UPNP_INLINE FILE *UpnpGetDebugFile_Inlined(
00180         Upnp_LogLevel level, Dbg_Module module)
00181 {
00182         (void)level;
00183         (void)module;
00184         return NULL;
00185 }
00186 #endif
00187
00192 void UpnpPrintf(
00195         Upnp_LogLevel DLevel,
00197         Dbg_Module Module,
00199         const char *DbgFileName,
00202         int DbgLineNo,
00204         const char *FmtStr,
00207         ...)
00208 #if (__GNUC__ >= 3)
00209         /* This enables printf like format checking by the compiler. */
00210         __attribute__((format(__printf__, 5, 6)))
00211 #endif
00212         ;
00213
00214 #if defined NDEBUG && !defined UPNP_DEBUG_C
00215         #define UpnpPrintf UpnpPrintf_Inlined
00216 // static UPNP_INLINE void UpnpPrintf_Inlined(Upnp_LogLevel DLevel,
00217 //      Dbg_Module Module,
00218 //      const char *DbgFileName,
00219 //      int DbgLineNo,
00220 //      const char *FmtStr,
00221 //      ...)
00222 // #if (__GNUC__ >= 3)
00223 //      /* This enables printf like format checking by the compiler. */
00224 //      __attribute__((format(__printf__, 5, 6)))
00225 // #endif
00226 //      ;
00227 static UPNP_INLINE void UpnpPrintf_Inlined(Upnp_LogLevel DLevel,
00228         Dbg_Module Module,
00229         const char *DbgFileName,
00230         int DbgLineNo,
00231         const char *FmtStr,
00232         ...)
00233 {
00234         (void)DLevel;
00235         (void)Module;
00236         (void)DbgFileName;
00237         (void)DbgLineNo;
00238         (void)FmtStr;
00239         return;
00240 }
00241 #endif /* DEBUG */
```

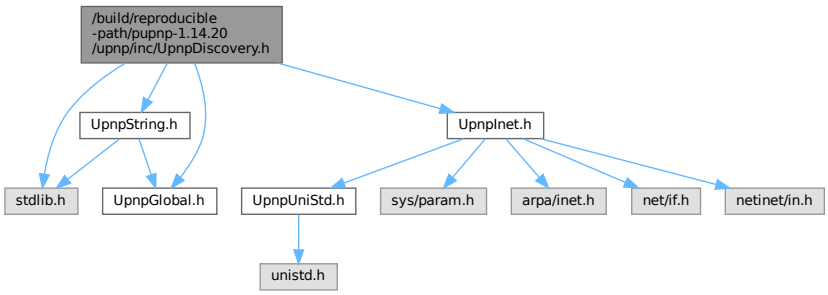
```
00242
00243 #ifdef __cplusplus
00244 }
00245 #endif
00246
00247 #endif /* UPNP_DEBUG_H */
```

9.37 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpDiscovery.h File Reference

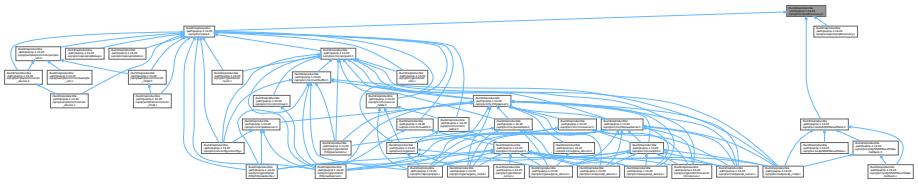
Header file for UpnpDiscovery methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include "UpnpString.h"
```

Include dependency graph for UpnpDiscovery.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpDiscovery UpnpDiscovery

Functions

- UpnpDiscovery * UpnpDiscovery_new ()
- void UpnpDiscovery_delete (UpnpDiscovery *p)
- UpnpDiscovery * UpnpDiscovery_dup (const UpnpDiscovery *p)
- int UpnpDiscovery_assign (UpnpDiscovery *p, const UpnpDiscovery *q)
- int UpnpDiscovery_get_ErrCode (const UpnpDiscovery *p)

- int UpnpDiscovery_set_ErrCode (UpnpDiscovery *p, int n)
- int UpnpDiscovery_get_Expires (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Expires (UpnpDiscovery *p, int n)
- const UpnpString * UpnpDiscovery_get_DeviceID (const UpnpDiscovery *p)
- int UpnpDiscovery_set_DeviceID (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_DeviceID_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_DeviceID_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_DeviceID (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_DeviceID (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_DeviceID (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_DeviceType (const UpnpDiscovery *p)
- int UpnpDiscovery_set_DeviceType (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_DeviceType_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_DeviceType_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_DeviceType (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_DeviceType (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_DeviceType (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_ServiceType (const UpnpDiscovery *p)
- int UpnpDiscovery_set_ServiceType (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_ServiceType_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_ServiceType_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_ServiceType (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_ServiceType (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_ServiceType (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_ServiceVer (const UpnpDiscovery *p)
- int UpnpDiscovery_set_ServiceVer (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_ServiceVer_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_ServiceVer_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_ServiceVer (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_ServiceVer (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_ServiceVer (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Location (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Location (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Location_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_Location_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Location (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Location (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Location (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Os (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Os (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Os_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_Os_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Os (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Os (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Os (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Date (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Date (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Date_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_Date_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Date (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Date (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Date (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Ext (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Ext (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Ext_Length (const UpnpDiscovery *p)

- const char * UpnpDiscovery_get_Ext_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Ext (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Ext (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Ext (UpnpDiscovery *p)
- const struct sockaddr_storage * UpnpDiscovery_get_DestAddr (const UpnpDiscovery *p)
- int UpnpDiscovery_set_DestAddr (UpnpDiscovery *p, const struct sockaddr_storage *buf)
- void UpnpDiscovery_clear_DestAddr (UpnpDiscovery *p)

9.37.1 Detailed Description

Header file for UpnpDiscovery methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.37.2 Typedef Documentation

9.37.2.1 UpnpDiscovery

```
typedef struct s_UpnpDiscovery UpnpDiscovery
```

UpnpDiscovery

9.37.3 Function Documentation

9.37.3.1 UpnpDiscovery_assign()

```
int UpnpDiscovery_assign (
            UpnpDiscovery * p,
            const UpnpDiscovery * q )
```

Assignment operator

References UpnpDiscovery_get_Date(), UpnpDiscovery_get_DestAddr(), UpnpDiscovery_get_DeviceID(), UpnpDiscovery_get_DeviceType(), UpnpDiscovery_get_ErrCode(), UpnpDiscovery_get_Expires(), UpnpDiscovery_get_Ext(), UpnpDiscovery_get_Location(), UpnpDiscovery_get_Os(), UpnpDiscovery_get_ServiceType(), UpnpDiscovery_get_ServiceVer(), UpnpDiscovery_set_Date(), UpnpDiscovery_set_DestAddr(), UpnpDiscovery_set_DeviceID(), UpnpDiscovery_set_DeviceType(), UpnpDiscovery_set_ErrCode(), UpnpDiscovery_set_Expires(), UpnpDiscovery_set_Ext(), UpnpDiscovery_set_Location(), UpnpDiscovery_set_Os(), UpnpDiscovery_set_ServiceType(), and UpnpDiscovery_set_ServiceVer().

Referenced by UpnpDiscovery_dup().

9.37.3.2 UpnpDiscovery_clear_Date()

```
void UpnpDiscovery_clear_Date (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Date

References UpnpString_clear().

9.37.3.3 UpnpDiscovery_clear_DestAddr()

```
void UpnpDiscovery_clear_DestAddr (
            UpnpDiscovery * p )
```

UpnpDiscovery_get_DestAddr

9.37.3.4 UpnpDiscovery_clear_DeviceID()

```
void UpnpDiscovery_clear_DeviceID (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_DeviceID

References UpnpString_clear().

9.37.3.5 UpnpDiscovery_clear_DeviceType()

```
void UpnpDiscovery_clear_DeviceType (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_DeviceType

References UpnpString_clear().

9.37.3.6 UpnpDiscovery_clear_Ext()

```
void UpnpDiscovery_clear_Ext (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Ext

References UpnpString_clear().

9.37.3.7 UpnpDiscovery_clear_Location()

```
void UpnpDiscovery_clear_Location (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Location

References UpnpString_clear().

9.37.3.8 UpnpDiscovery_clear_Os()

```
void UpnpDiscovery_clear_Os (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Os

References UpnpString_clear().

9.37.3.9 UpnpDiscovery_clear_ServiceType()

```
void UpnpDiscovery_clear_ServiceType (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_ServiceType

References UpnpString_clear().

9.37.3.10 UpnpDiscovery_clear_ServiceVer()

```
void UpnpDiscovery_clear_ServiceVer (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_ServiceVer

References UpnpString_clear().

9.37.3.11 UpnpDiscovery_delete()

```
void UpnpDiscovery_delete (
            UpnpDiscovery * p )
```

Destructor

References UpnpString_delete().

Referenced by ssdp_handle_ctrlpt_msg(), SSDPResultData_delete(), and SSDPResultData_set_Param().

9.37.3.12 UpnpDiscovery_dup()

```
UpnpDiscovery * UpnpDiscovery_dup (
            const UpnpDiscovery * p )
```

Copy Constructor

References UpnpDiscovery_assign(), and UpnpDiscovery_new().

Referenced by SSDPResultData_set_Param().

9.37.3.13 UpnpDiscovery_get_Date()

```
const UpnpString * UpnpDiscovery_get_Date (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Date

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_Date_cstr(), and UpnpDiscovery_get_Date_Length().

9.37.3.14 UpnpDiscovery_get_Date_cstr()

```
const char * UpnpDiscovery_get_Date_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Date_cstr

References UpnpDiscovery_get_Date(), and UpnpString_get_String().

9.37.3.15 UpnpDiscovery_get_Date_Length()

```
size_t UpnpDiscovery_get_Date_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Date_Length

References UpnpDiscovery_get_Date(), and UpnpString_get_Length().

9.37.3.16 UpnpDiscovery_get_DestAddr()

```
const struct sockaddr_storage * UpnpDiscovery_get_DestAddr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DestAddr

Referenced by UpnpDiscovery_assign().

9.37.3.17 UpnpDiscovery_get_DeviceID()

```
const UpnpString * UpnpDiscovery_get_DeviceID (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceID

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_DeviceID_cstr(), and UpnpDiscovery_get_DeviceID_Length().

9.37.3.18 UpnpDiscovery_get_DeviceID_cstr()

```
const char * UpnpDiscovery_get_DeviceID_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceID_cstr

References UpnpDiscovery_get_DeviceID(), and UpnpString_get_String().

9.37.3.19 UpnpDiscovery_get_DeviceID_Length()

```
size_t UpnpDiscovery_get_DeviceID_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceID_Length

References UpnpDiscovery_get_DeviceID(), and UpnpString_get_Length().

9.37.3.20 UpnpDiscovery_get_DeviceType()

```
const UpnpString * UpnpDiscovery_get_DeviceType (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceType

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_DeviceType_cstr(), and UpnpDiscovery_get_DeviceType_Length().

9.37.3.21 UpnpDiscovery_get_DeviceType_cstr()

```
const char * UpnpDiscovery_get_DeviceType_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceType_cstr

References UpnpDiscovery_get_DeviceType(), and UpnpString_get_String().

9.37.3.22 UpnpDiscovery_get_DeviceType_Length()

```
size_t UpnpDiscovery_get_DeviceType_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceType_Length

References UpnpDiscovery_get_DeviceType(), and UpnpString_get_Length().

9.37.3.23 UpnpDiscovery_get_ErrCode()

```
int UpnpDiscovery_get_ErrCode (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpDiscovery_assign().

9.37.3.24 UpnpDiscovery_get_Expires()

```
int UpnpDiscovery_get_Expires (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Expires

Referenced by SampleUtil_PrintEvent(), ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.37.3.25 UpnpDiscovery_get_Ext()

```
const UpnpString * UpnpDiscovery_get_Ext (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Ext

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_Ext_cstr(), and UpnpDiscovery_get_Ext_Leng

9.37.3.26 UpnpDiscovery_get_Ext_cstr()

```
const char * UpnpDiscovery_get_Ext_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Ext_cstr

References UpnpDiscovery_get_Ext(), and UpnpString_get_String().

9.37.3.27 UpnpDiscovery_get_Ext_Length()

```
size_t UpnpDiscovery_get_Ext_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Ext_Length

References UpnpDiscovery_get_Ext(), and UpnpString_get_Length().

9.37.3.28 UpnpDiscovery_get_Location()

```
const UpnpString * UpnpDiscovery_get_Location (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Location

Referenced by SampleUtil_PrintEvent(), ssdp_handle_ctrlpt_msg(), UpnpDiscovery_assign(), UpnpDiscovery_get_Location_cstr(), and UpnpDiscovery_get_Location_Length().

9.37.3.29 UpnpDiscovery_get_Location_cstr()

```
const char * UpnpDiscovery_get_Location_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Location_cstr

References UpnpDiscovery_get_Location(), and UpnpString_get_String().

9.37.3.30 UpnpDiscovery_get_Location_Length()

```
size_t UpnpDiscovery_get_Location_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Location_Length

References UpnpDiscovery_get_Location(), and UpnpString_get_Length().

9.37.3.31 UpnpDiscovery_get_Os()

```
const UpnpString * UpnpDiscovery_get_Os (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Os

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_Os_cstr(), and UpnpDiscovery_get_Os_Lengt

9.37.3.32 UpnpDiscovery_get_Os_cstr()

```
const char * UpnpDiscovery_get_Os_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Os_cstr

References UpnpDiscovery_get_Os(), and UpnpString_get_String().

9.37.3.33 UpnpDiscovery_get_Os_Length()

```
size_t UpnpDiscovery_get_Os_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Os_Length

References UpnpDiscovery_get_Os(), and UpnpString_get_Length().

9.37.3.34 UpnpDiscovery_get_ServiceType()

```
const UpnpString * UpnpDiscovery_get_ServiceType (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceType

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_ServiceType_cstr(), and UpnpDiscovery_get_ServiceType_Length().

9.37.3.35 UpnpDiscovery_get_ServiceType_cstr()

```
const char * UpnpDiscovery_get_ServiceType_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceType_cstr

References UpnpDiscovery_get_ServiceType(), and UpnpString_get_String().

9.37.3.36 UpnpDiscovery_get_ServiceType_Length()

```
size_t UpnpDiscovery_get_ServiceType_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceType_Length

References UpnpDiscovery_get_ServiceType(), and UpnpString_get_Length().

9.37.3.37 UpnpDiscovery_get_ServiceVer()

```
const UpnpString * UpnpDiscovery_get_ServiceVer (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceVer

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_ServiceVer_cstr(), and UpnpDiscovery_get_ServiceVer_Length().

9.37.3.38 UpnpDiscovery_get_ServiceVer_cstr()

```
const char * UpnpDiscovery_get_ServiceVer_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceVer_cstr

References UpnpDiscovery_get_ServiceVer(), and UpnpString_get_String().

9.37.3.39 UpnpDiscovery_get_ServiceVer_Length()

```
size_t UpnpDiscovery_get_ServiceVer_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceVer_Length

References UpnpDiscovery_get_ServiceVer(), and UpnpString_get_Length().

9.37.3.40 UpnpDiscovery_new()

```
UpnpDiscovery * UpnpDiscovery_new ( )
```

Constructor

References UpnpString_new().

Referenced by ssdp_handle_ctrlpt_msg(), SSDPResultData_new(), and UpnpDiscovery_dup().

9.37.3.41 UpnpDiscovery_set_Date()

```
int UpnpDiscovery_set_Date (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Date

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.42 UpnpDiscovery_set_DestAddr()

```
int UpnpDiscovery_set_DestAddr (
            UpnpDiscovery * p,
            const struct sockaddr_storage * buf )
```

UpnpDiscovery_get_DestAddr

Referenced by ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.37.3.43 UpnpDiscovery_set_DeviceID()

```
int UpnpDiscovery_set_DeviceID (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_DeviceID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.44 UpnpDiscovery_set_DeviceType()

```
int UpnpDiscovery_set_DeviceType (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_DeviceType

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.45 UpnpDiscovery_set_ErrCode()

```
int UpnpDiscovery_set_ErrCode (
            UpnpDiscovery * p,
            int n )
```

UpnpDiscovery_set_ErrCode

Referenced by ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.37.3.46 UpnpDiscovery_set_Expires()

```
int UpnpDiscovery_set_Expires (
            UpnpDiscovery * p,
            int n )
```

UpnpDiscovery_set_Expires

Referenced by ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.37.3.47 UpnpDiscovery_set_Ext()

```
int UpnpDiscovery_set_Ext (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Ext

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.48 UpnpDiscovery_set_Location()

```
int UpnpDiscovery_set_Location (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Location

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.49 UpnpDiscovery_set_Os()

```
int UpnpDiscovery_set_Os (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Os

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.50 UpnpDiscovery_set_ServiceType()

```
int UpnpDiscovery_set_ServiceType (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_ServiceType

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.51 UpnpDiscovery_set_ServiceVer()

```
int UpnpDiscovery_set_ServiceVer (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_ServiceVer

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.37.3.52 UpnpDiscovery_strcpy_Date()

```
int UpnpDiscovery_strcpy_Date (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Date

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.37.3.53 UpnpDiscovery_strcpy_DeviceID()

```
int UpnpDiscovery_strcpy_DeviceID (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_DeviceID

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.37.3.54 UpnpDiscovery_strcpy_DeviceType()

```
int UpnpDiscovery_strcpy_DeviceType (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_DeviceType

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.37.3.55 UpnpDiscovery_strcpy_Ext()

```
int UpnpDiscovery_strcpy_Ext (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Ext

References UpnpString_set_String().

9.37.3.56 UpnpDiscovery_strcpy_Location()

```
int UpnpDiscovery_strcpy_Location (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Location

References UpnpString_set_String().

9.37.3.57 UpnpDiscovery_strcpy_Os()

```
int UpnpDiscovery_strcpy_Os (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Os

References UpnpString_set_String().

9.37.3.58 UpnpDiscovery_strcpy_ServiceType()

```
int UpnpDiscovery_strcpy_ServiceType (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_ServiceType

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.37.3.59 UpnpDiscovery_strcpy_ServiceVer()

```
int UpnpDiscovery_strcpy_ServiceVer (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_ServiceVer

References UpnpString_set_String().

9.37.3.60 UpnpDiscovery_strncpy_Date()

```
int UpnpDiscovery_strncpy_Date (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Date

References UpnpString_set_StringN().

9.37.3.61 UpnpDiscovery_strncpy_DeviceID()

```
int UpnpDiscovery_strncpy_DeviceID (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_DeviceID

References UpnpString_set_StringN().

9.37.3.62 UpnpDiscovery_strncpy_DeviceType()

```
int UpnpDiscovery_strncpy_DeviceType (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_DeviceType

References UpnpString_set_StringN().

9.37.3.63 UpnpDiscovery_strncpy_Ext()

```
int UpnpDiscovery_strncpy_Ext (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Ext

References UpnpString_set_StringN().

Referenced by ssdp_handle_ctrlpt_msg().

9.37.3.64 UpnpDiscovery_strncpy_Location()

```
int UpnpDiscovery_strncpy_Location (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Location

References UpnpString_set_StringN().

Referenced by ssdp_handle_ctrlpt_msg().

9.37.3.65 UpnpDiscovery_strncpy_Os()

```
int UpnpDiscovery_strncpy_Os (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Os

References UpnpString_set_StringN().

Referenced by ssdp_handle_ctrlpt_msg().

9.37.3.66 UpnpDiscovery_strncpy_ServiceType()

```
int UpnpDiscovery_strncpy_ServiceType (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_ServiceType

References UpnpString_set_StringN().

9.37.3.67 UpnpDiscovery_strncpy_ServiceVer()

```
int UpnpDiscovery_strncpy_ServiceVer (
             UpnpDiscovery * p,
             const char * s,
             size_t n )
```

UpnpDiscovery_strncpy_ServiceVer

References UpnpString_set_StringN().

9.38 UpnpDiscovery.h

Go to the documentation of this file.

```
00001 #ifndef UPNPDISCOVERY_H
00002 #define UPNPDISCOVERY_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpInet.h"
00019 #include "UpnpString.h"
00020
00021 #ifdef __cplusplus
00022 extern "C" {
00023 #endif /* __cplusplus */
00024
00028 typedef struct s_UpnpDiscovery UpnpDiscovery;
00029
00031 UPNP_EXPORT_SPEC UpnpDiscovery *UpnpDiscovery_new();
00033 UPNP_EXPORT_SPEC void UpnpDiscovery_delete(UpnpDiscovery *p);
00035 UPNP_EXPORT_SPEC UpnpDiscovery *UpnpDiscovery_dup(const UpnpDiscovery *p);
00037 UPNP_EXPORT_SPEC int UpnpDiscovery_assign(
00038         UpnpDiscovery *p, const UpnpDiscovery *q);
00039
00041 UPNP_EXPORT_SPEC int UpnpDiscovery_get_ErrCode(const UpnpDiscovery *p);
00043 UPNP_EXPORT_SPEC int UpnpDiscovery_set_ErrCode(UpnpDiscovery *p, int n);
00044
00046 UPNP_EXPORT_SPEC int UpnpDiscovery_get_Expires(const UpnpDiscovery *p);
00048 UPNP_EXPORT_SPEC int UpnpDiscovery_set_Expires(UpnpDiscovery *p, int n);
00049
00051 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_DeviceID(
00052         const UpnpDiscovery *p);
00054 UPNP_EXPORT_SPEC int UpnpDiscovery_set_DeviceID(
00055         UpnpDiscovery *p, const UpnpString *s);
00057 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_DeviceID_Length(
00058         const UpnpDiscovery *p);
00060 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_DeviceID_cstr(
00061         const UpnpDiscovery *p);
00063 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_DeviceID(
00064         UpnpDiscovery *p, const char *s);
00066 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_DeviceID(
00067         UpnpDiscovery *p, const char *s, size_t n);
00069 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_DeviceID(UpnpDiscovery *p);
00070
00072 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_DeviceType(
00073         const UpnpDiscovery *p);
00075 UPNP_EXPORT_SPEC int UpnpDiscovery_set_DeviceType(
00076         UpnpDiscovery *p, const UpnpString *s);
00078 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_DeviceType_Length(
00079         const UpnpDiscovery *p);
00081 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_DeviceType_cstr(
00082         const UpnpDiscovery *p);
00084 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_DeviceType(
00085         UpnpDiscovery *p, const char *s);
00087 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_DeviceType(
00088         UpnpDiscovery *p, const char *s, size_t n);
00090 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_DeviceType(UpnpDiscovery *p);
00091
00093 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_ServiceType(
00094         const UpnpDiscovery *p);
00096 UPNP_EXPORT_SPEC int UpnpDiscovery_set_ServiceType(
00097         UpnpDiscovery *p, const UpnpString *s);
00099 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_ServiceType_Length(
00100         const UpnpDiscovery *p);
```

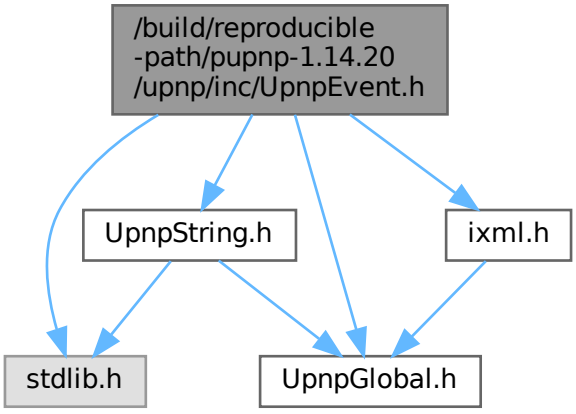
```
00102 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_ServiceType_cstr(
00103         const UpnpDiscovery *p);
00105 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_ServiceType(
00106         UpnpDiscovery *p, const char *s);
00108 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_ServiceType(
00109         UpnpDiscovery *p, const char *s, size_t n);
00111 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_ServiceType(UpnpDiscovery *p);
00112
00114 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_ServiceVer(
00115         const UpnpDiscovery *p);
00117 UPNP_EXPORT_SPEC int UpnpDiscovery_set_ServiceVer(
00118         UpnpDiscovery *p, const UpnpString *s);
00120 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_ServiceVer_Length(
00121         const UpnpDiscovery *p);
00123 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_ServiceVer_cstr(
00124         const UpnpDiscovery *p);
00126 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_ServiceVer(
00127         UpnpDiscovery *p, const char *s);
00129 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_ServiceVer(
00130         UpnpDiscovery *p, const char *s, size_t n);
00132 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_ServiceVer(UpnpDiscovery *p);
00133
00135 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_Location(
00136         const UpnpDiscovery *p);
00138 UPNP_EXPORT_SPEC int UpnpDiscovery_set_Location(
00139         UpnpDiscovery *p, const UpnpString *s);
00141 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_Location_Length(
00142         const UpnpDiscovery *p);
00144 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_Location_cstr(
00145         const UpnpDiscovery *p);
00147 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_Location(
00148         UpnpDiscovery *p, const char *s);
00150 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_Location(
00151         UpnpDiscovery *p, const char *s, size_t n);
00153 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_Location(UpnpDiscovery *p);
00154
00156 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_Os(const UpnpDiscovery *p);
00158 UPNP_EXPORT_SPEC int UpnpDiscovery_set_Os(
00159         UpnpDiscovery *p, const UpnpString *s);
00161 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_Os_Length(const UpnpDiscovery *p);
00163 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_Os_cstr(const UpnpDiscovery *p);
00165 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_Os(UpnpDiscovery *p, const char *s);
00167 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_Os(
00168         UpnpDiscovery *p, const char *s, size_t n);
00170 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_Os(UpnpDiscovery *p);
00171
00173 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_Date(
00174         const UpnpDiscovery *p);
00176 UPNP_EXPORT_SPEC int UpnpDiscovery_set_Date(
00177         UpnpDiscovery *p, const UpnpString *s);
00179 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_Date_Length(const UpnpDiscovery *p);
00181 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_Date_cstr(
00182         const UpnpDiscovery *p);
00184 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_Date(UpnpDiscovery *p, const char *s);
00186 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_Date(
00187         UpnpDiscovery *p, const char *s, size_t n);
00189 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_Date(UpnpDiscovery *p);
00190
00192 UPNP_EXPORT_SPEC const UpnpString *UpnpDiscovery_get_Ext(
00193         const UpnpDiscovery *p);
00195 UPNP_EXPORT_SPEC int UpnpDiscovery_set_Ext(
00196         UpnpDiscovery *p, const UpnpString *s);
00198 UPNP_EXPORT_SPEC size_t UpnpDiscovery_get_Ext_Length(const UpnpDiscovery *p);
00200 UPNP_EXPORT_SPEC const char *UpnpDiscovery_get_Ext_cstr(const UpnpDiscovery *p);
00202 UPNP_EXPORT_SPEC int UpnpDiscovery_strcpy_Ext(UpnpDiscovery *p, const char *s);
00204 UPNP_EXPORT_SPEC int UpnpDiscovery_strncpy_Ext(
00205         UpnpDiscovery *p, const char *s, size_t n);
00207 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_Ext(UpnpDiscovery *p);
00208
00210 UPNP_EXPORT_SPEC const struct sockaddr_storage *UpnpDiscovery_get_DestAddr(
00211         const UpnpDiscovery *p);
00213 UPNP_EXPORT_SPEC int UpnpDiscovery_set_DestAddr(
00214         UpnpDiscovery *p, const struct sockaddr_storage *buf);
00216 UPNP_EXPORT_SPEC void UpnpDiscovery_clear_DestAddr(UpnpDiscovery *p);
00217
00218 #ifdef __cplusplus
00219 }
00220 #endif /* __cplusplus */
00221
00222 #endif /* UPNPDISCOVERY_H */
```

9.39 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEvent.h File Reference

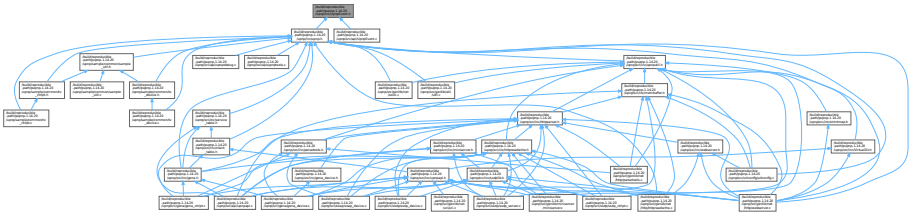
Header file for UpnpEvent methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpString.h"
#include "ixml.h"
```

Include dependency graph for UpnpEvent.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpEvent UpnpEvent

Functions

- UpnpEvent * UpnpEvent_new ()
- void UpnpEvent_delete (UpnpEvent *p)
- UpnpEvent * UpnpEvent_dup (const UpnpEvent *p)
- int UpnpEvent_assign (UpnpEvent *p, const UpnpEvent *q)

- int UpnpEvent_get_EventKey (const UpnpEvent *p)
- int UpnpEvent_set_EventKey (UpnpEvent *p, int n)
- IXML_Document * UpnpEvent_get_ChangedVariables (const UpnpEvent *p)
- int UpnpEvent_set_ChangedVariables (UpnpEvent *p, IXML_Document *n)
- const UpnpString * UpnpEvent_get_SID (const UpnpEvent *p)
- int UpnpEvent_set_SID (UpnpEvent *p, const UpnpString *s)
- size_t UpnpEvent_get_SID_Length (const UpnpEvent *p)
- const char * UpnpEvent_get_SID_cstr (const UpnpEvent *p)
- int UpnpEvent_strcpy_SID (UpnpEvent *p, const char *s)
- int UpnpEvent_strncpy_SID (UpnpEvent *p, const char *s, size_t n)
- void UpnpEvent_clear_SID (UpnpEvent *p)

9.39.1 Detailed Description

Header file for UpnpEvent methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.39.2 Typedef Documentation

9.39.2.1 UpnpEvent

```
typedef struct s_UpnpEvent UpnpEvent
```

UpnpEvent

9.39.3 Function Documentation

9.39.3.1 UpnpEvent_assign()

```
int UpnpEvent_assign (
            UpnpEvent * p,
            const UpnpEvent * q )
```

Assignment operator

References UpnpEvent_get_ChangedVariables(), UpnpEvent_get_EventKey(), UpnpEvent_get_SID(), UpnpEvent_set_ChangedVaria
UpnpEvent_set_EventKey(), and UpnpEvent_set_SID().

Referenced by UpnpEvent_dup().

9.39.3.2 UpnpEvent_clear_SID()

```
void UpnpEvent_clear_SID (
            UpnpEvent * p )
```

UpnpEvent_clear_SID

References UpnpString_clear().

9.39.3.3 UpnpEvent_delete()

```
void UpnpEvent_delete (
            UpnpEvent * p )
```

Destructor

References UpnpString_delete().

Referenced by gena_process_notification_event().

9.39.3.4 UpnpEvent_dup()

```
UpnpEvent * UpnpEvent_dup (
            const UpnpEvent * p )
```

Copy Constructor

References UpnpEvent_assign(), and UpnpEvent_new().

9.39.3.5 UpnpEvent_get_ChangedVariables()

```
IXML_Document * UpnpEvent_get_ChangedVariables (
            const UpnpEvent * p )
```

UpnpEvent_get_ChangedVariables

Referenced by SampleUtil_PrintEvent(), and UpnpEvent_assign().

9.39.3.6 UpnpEvent_get_EventKey()

```
int UpnpEvent_get_EventKey (
            const UpnpEvent * p )
```

UpnpEvent_get_EventKey

Referenced by SampleUtil_PrintEvent(), and UpnpEvent_assign().

9.39.3.7 UpnpEvent_get_SID()

```
const UpnpString * UpnpEvent_get_SID (
            const UpnpEvent * p )
```

UpnpEvent_get_SID

Referenced by SampleUtil_PrintEvent(), UpnpEvent_assign(), UpnpEvent_get_SID_cstr(), and UpnpEvent_get_SID_Length().

9.39.3.8 UpnpEvent_get_SID_cstr()

```
const char * UpnpEvent_get_SID_cstr (
            const UpnpEvent * p )
```

UpnpEvent_get_SID_cstr

References UpnpEvent_get_SID(), and UpnpString_get_String().

9.39.3.9 UpnpEvent_get_SID_Length()

```
size_t UpnpEvent_get_SID_Length (
            const UpnpEvent * p )
```

UpnpEvent_get_SID_Length

References UpnpEvent_get_SID(), and UpnpString_get_Length().

9.39.3.10 UpnpEvent_new()

```
UpnpEvent * UpnpEvent_new ( )
```

Constructor

References UpnpString_new().

Referenced by gena_process_notification_event(), and UpnpEvent_dup().

9.39.3.11 UpnpEvent_set_ChangedVariables()

```
int UpnpEvent_set_ChangedVariables (
            UpnpEvent * p,
            IXML_Document * n )
```

UpnpEvent_set_ChangedVariables

Referenced by gena_process_notification_event(), and UpnpEvent_assign().

9.39.3.12 UpnpEvent_set_EventKey()

```
int UpnpEvent_set_EventKey (
            UpnpEvent * p,
            int n )
```

UpnpEvent_set_EventKey

Referenced by gena_process_notification_event(), and UpnpEvent_assign().

9.39.3.13 UpnpEvent_set_SID()

```
int UpnpEvent_set_SID (
            UpnpEvent * p,
            const UpnpString * s )
```

UpnpEvent_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by gena_process_notification_event(), and UpnpEvent_assign().

9.39.3.14 UpnpEvent_strcpy_SID()

```
int UpnpEvent_strcpy_SID (
            UpnpEvent * p,
            const char * s )
```

UpnpEvent_strcpy_SID

References UpnpString_set_String().

9.39.3.15 UpnpEvent_strncpy_SID()

```
int UpnpEvent_strncpy_SID (
            UpnpEvent * p,
            const char * s,
            size_t n )
```

UpnpEvent_strncpy_SID

References UpnpString_set_StringN().

9.40 UpnpEvent.h

Go to the documentation of this file.

```
00001 #ifndef UPNPEVENT_H
00002 #define UPNPEVENT_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpString.h"
00019 #include "ixml.h"
00020
00021 #ifdef __cplusplus
00022 extern "C" {
00023 #endif /* __cplusplus */
00024
00028 typedef struct s_UpnpEvent UpnpEvent;
00029
00031 UPNP_EXPORT_SPEC UpnpEvent *UpnpEvent_new();
00033 UPNP_EXPORT_SPEC void UpnpEvent_delete(UpnpEvent *p);
00035 UPNP_EXPORT_SPEC UpnpEvent *UpnpEvent_dup(const UpnpEvent *p);
00037 UPNP_EXPORT_SPEC int UpnpEvent_assign(UpnpEvent *p, const UpnpEvent *q);
00038
00040 UPNP_EXPORT_SPEC int UpnpEvent_get_EventKey(const UpnpEvent *p);
00042 UPNP_EXPORT_SPEC int UpnpEvent_set_EventKey(UpnpEvent *p, int n);
00043
```

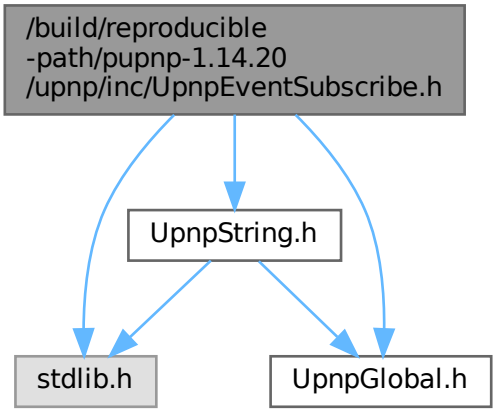
```
00045 UPNP_EXPORT_SPEC IXML_Document *UpnpEvent_get_ChangedVariables(
00046         const UpnpEvent *p);
00048 UPNP_EXPORT_SPEC int UpnpEvent_set_ChangedVariables(
00049         UpnpEvent *p, IXML_Document *n);
00050
00052 UPNP_EXPORT_SPEC const UpnpString *UpnpEvent_get_SID(const UpnpEvent *p);
00054 UPNP_EXPORT_SPEC int UpnpEvent_set_SID(UpnpEvent *p, const UpnpString *s);
00056 UPNP_EXPORT_SPEC size_t UpnpEvent_get_SID_Length(const UpnpEvent *p);
00058 UPNP_EXPORT_SPEC const char *UpnpEvent_get_SID_cstr(const UpnpEvent *p);
00060 UPNP_EXPORT_SPEC int UpnpEvent_strcpy_SID(UpnpEvent *p, const char *s);
00062 UPNP_EXPORT_SPEC int UpnpEvent_strncpy_SID(
00063         UpnpEvent *p, const char *s, size_t n);
00065 UPNP_EXPORT_SPEC void UpnpEvent_clear_SID(UpnpEvent *p);
00066
00067 #ifdef __cplusplus
00068 }
00069 #endif /* __cplusplus */
00070
00071 #endif /* UPNPEVENT_H */
```

9.41 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEventSubscribe.h File Reference

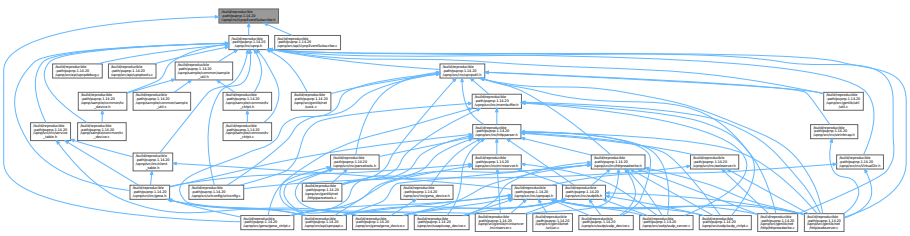
Header file for UpnpEventSubscribe methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpString.h"
```

Include dependency graph for UpnpEventSubscribe.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpEventSubscribe UpnpEventSubscribe

Functions

- UpnpEventSubscribe * UpnpEventSubscribe_new ()
- void UpnpEventSubscribe_delete (UpnpEventSubscribe *p)
- UpnpEventSubscribe * UpnpEventSubscribe_dup (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_assign (UpnpEventSubscribe *p, const UpnpEventSubscribe *q)
- int UpnpEventSubscribe_get_ErrCode (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_ErrCode (UpnpEventSubscribe *p, int n)
- int UpnpEventSubscribe_get_TimeOut (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_TimeOut (UpnpEventSubscribe *p, int n)
- const UpnpString * UpnpEventSubscribe_get_SID (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_SID (UpnpEventSubscribe *p, const UpnpString *s)
- size_t UpnpEventSubscribe_get_SID_Length (const UpnpEventSubscribe *p)
- const char * UpnpEventSubscribe_get_SID_cstr (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_strcpy_SID (UpnpEventSubscribe *p, const char *s)
- int UpnpEventSubscribe_strncpy_SID (UpnpEventSubscribe *p, const char *s, size_t n)
- void UpnpEventSubscribe_clear_SID (UpnpEventSubscribe *p)
- const UpnpString * UpnpEventSubscribe_get_PublisherUrl (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_PublisherUrl (UpnpEventSubscribe *p, const UpnpString *s)
- size_t UpnpEventSubscribe_get_PublisherUrl_Length (const UpnpEventSubscribe *p)
- const char * UpnpEventSubscribe_get_PublisherUrl_cstr (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_strcpy_PublisherUrl (UpnpEventSubscribe *p, const char *s)
- int UpnpEventSubscribe_strncpy_PublisherUrl (UpnpEventSubscribe *p, const char *s, size_t n)
- void UpnpEventSubscribe_clear_PublisherUrl (UpnpEventSubscribe *p)

9.41.1 Detailed Description

Header file for UpnpEventSubscribe methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.41.2 Typedef Documentation

9.41.2.1 UpnpEventSubscribe

```
typedef struct s_UpnpEventSubscribe UpnpEventSubscribe
```

UpnpEventSubscribe

9.41.3 Function Documentation

9.41.3.1 UpnpEventSubscribe_assign()

```
int UpnpEventSubscribe_assign (
            UpnpEventSubscribe * p,
            const UpnpEventSubscribe * q )
```

Assignment operator

References UpnpEventSubscribe_get_ErrCode(), UpnpEventSubscribe_get_PublisherUrl(), UpnpEventSubscribe_get_SID(), UpnpEventSubscribe_get_TimeOut(), UpnpEventSubscribe_set_ErrCode(), UpnpEventSubscribe_set_PublisherUrl(), UpnpEventSubscribe_set_SID(), and UpnpEventSubscribe_set_TimeOut().

Referenced by UpnpEventSubscribe_dup().

9.41.3.2 UpnpEventSubscribe_clear_PublisherUrl()

```
void UpnpEventSubscribe_clear_PublisherUrl (
            UpnpEventSubscribe * p )
```

UpnpEventSubscribe_clear_PublisherUrl

References UpnpString_clear().

9.41.3.3 UpnpEventSubscribe_clear_SID()

```
void UpnpEventSubscribe_clear_SID (
            UpnpEventSubscribe * p )
```

UpnpEventSubscribe_clear_SID

References UpnpString_clear().

9.41.3.4 UpnpEventSubscribe_delete()

```
void UpnpEventSubscribe_delete (
            UpnpEventSubscribe * p )
```

Destructor

References UpnpString_delete().

Referenced by free_subscribe_arg(), and UpnpThreadDistribution().

9.41.3.5 UpnpEventSubscribe_dup()

```
UpnpEventSubscribe * UpnpEventSubscribe_dup (
            const UpnpEventSubscribe * p )
```

Copy Constructor

References UpnpEventSubscribe_assign(), and UpnpEventSubscribe_new().

9.41.3.6 UpnpEventSubscribe_get_ErrCode()

```
int UpnpEventSubscribe_get_ErrCode (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpEventSubscribe_assign().

9.41.3.7 UpnpEventSubscribe_get_PublisherUrl()

```
const UpnpString * UpnpEventSubscribe_get_PublisherUrl (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_PublisherUrl

Referenced by SampleUtil_PrintEvent(), UpnpEventSubscribe_assign(), UpnpEventSubscribe_get_PublisherUrl_cstr(), UpnpEventSubscribe_get_PublisherUrl_Length(), and UpnpThreadDistribution().

9.41.3.8 UpnpEventSubscribe_get_PublisherUrl_cstr()

```
const char * UpnpEventSubscribe_get_PublisherUrl_cstr (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_PublisherUrl_cstr

References UpnpEventSubscribe_get_PublisherUrl(), and UpnpString_get_String().

9.41.3.9 UpnpEventSubscribe_get_PublisherUrl_Length()

```
size_t UpnpEventSubscribe_get_PublisherUrl_Length (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_PublisherUrl_Length

References UpnpEventSubscribe_get_PublisherUrl(), and UpnpString_get_Length().

9.41.3.10 UpnpEventSubscribe_get_SID()

```
const UpnpString * UpnpEventSubscribe_get_SID (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_SID

Referenced by GenaAutoRenewSubscription(), SampleUtil_PrintEvent(), UpnpEventSubscribe_assign(), UpnpEventSubscribe_get_SID_cstr(), UpnpEventSubscribe_get_SID_Length(), and UpnpThreadDistribution().

9.41.3.11 UpnpEventSubscribe_get_SID_cstr()

```
const char * UpnpEventSubscribe_get_SID_cstr (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_SID_cstr

References UpnpEventSubscribe_get_SID(), and UpnpString_get_String().

9.41.3.12 UpnpEventSubscribe_get_SID_Length()

```
size_t UpnpEventSubscribe_get_SID_Length (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_SID_Length

References UpnpEventSubscribe_get_SID(), and UpnpString_get_Length().

9.41.3.13 UpnpEventSubscribe_get_TimeOut()

```
int UpnpEventSubscribe_get_TimeOut (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_TimeOut

Referenced by GenaAutoRenewSubscription(), SampleUtil_PrintEvent(), and UpnpEventSubscribe_assign().

9.41.3.14 UpnpEventSubscribe_new()

```
UpnpEventSubscribe * UpnpEventSubscribe_new ( )
```

Constructor

References UpnpString_new().

Referenced by ScheduleGenaAutoRenew(), UpnpEventSubscribe_dup(), and UpnpThreadDistribution().

9.41.3.15 UpnpEventSubscribe_set_ErrCode()

```
int UpnpEventSubscribe_set_ErrCode (
            UpnpEventSubscribe * p,
            int n )
```

UpnpEventSubscribe_set_ErrCode

Referenced by GenaAutoRenewSubscription(), ScheduleGenaAutoRenew(), UpnpEventSubscribe_assign(), and UpnpThreadDistribution().

9.41.3.16 UpnpEventSubscribe_set_PublisherUrl()

```
int UpnpEventSubscribe_set_PublisherUrl (
            UpnpEventSubscribe * p,
            const UpnpString * s )
```

UpnpEventSubscribe_set_PublisherUrl

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by ScheduleGenaAutoRenew(), and UpnpEventSubscribe_assign().

9.41.3.17 UpnpEventSubscribe_set_SID()

```
int UpnpEventSubscribe_set_SID (
            UpnpEventSubscribe * p,
            const UpnpString * s )
```

UpnpEventSubscribe_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by ScheduleGenaAutoRenew(), UpnpEventSubscribe_assign(), and UpnpThreadDistribution().

9.41.3.18 UpnpEventSubscribe_set_TimeOut()

```
int UpnpEventSubscribe_set_TimeOut (
            UpnpEventSubscribe * p,
            int n )
```

UpnpEventSubscribe_set_TimeOut

Referenced by GenaAutoRenewSubscription(), ScheduleGenaAutoRenew(), UpnpEventSubscribe_assign(), and UpnpThreadDistribution().

9.41.3.19 UpnpEventSubscribe_strcpy_PublisherUrl()

```
int UpnpEventSubscribe_strcpy_PublisherUrl (
            UpnpEventSubscribe * p,
            const char * s )
```

UpnpEventSubscribe_strcpy_PublisherUrl

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.41.3.20 UpnpEventSubscribe_strcpy_SID()

```
int UpnpEventSubscribe_strcpy_SID (
            UpnpEventSubscribe * p,
            const char * s )
```

UpnpEventSubscribe_strcpy_SID

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.41.3.21 UpnpEventSubscribe_strncpy_PublisherUrl()

```
int UpnpEventSubscribe_strncpy_PublisherUrl (
            UpnpEventSubscribe * p,
            const char * s,
            size_t n )
```

UpnpEventSubscribe_strncpy_PublisherUrl

References UpnpString_set_StringN().

9.41.3.22 UpnpEventSubscribe_strncpy_SID()

```
int UpnpEventSubscribe_strncpy_SID (
            UpnpEventSubscribe * p,
            const char * s,
            size_t n )
```

UpnpEventSubscribe_strncpy_SID

References UpnpString_set_StringN().

9.42 UpnpEventSubscribe.h

Go to the documentation of this file.

```
00001 #ifndef UPNPEVENTSUBSCRIBE_H
00002 #define UPNPEVENTSUBSCRIBE_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpString.h"
00019
00020 #ifdef __cplusplus
00021 extern "C" {
00022 #endif /* __cplusplus */
00023
00027 typedef struct s_UpnpEventSubscribe UpnpEventSubscribe;
00028
00030 UPNP_EXPORT_SPEC UpnpEventSubscribe *UpnpEventSubscribe_new();
00032 UPNP_EXPORT_SPEC void UpnpEventSubscribe_delete(UpnpEventSubscribe *p);
00034 UPNP_EXPORT_SPEC UpnpEventSubscribe *UpnpEventSubscribe_dup(
00035         const UpnpEventSubscribe *p);
00037 UPNP_EXPORT_SPEC int UpnpEventSubscribe_assign(
00038         UpnpEventSubscribe *p, const UpnpEventSubscribe *q);
00039
00041 UPNP_EXPORT_SPEC int UpnpEventSubscribe_get_ErrCode(
```

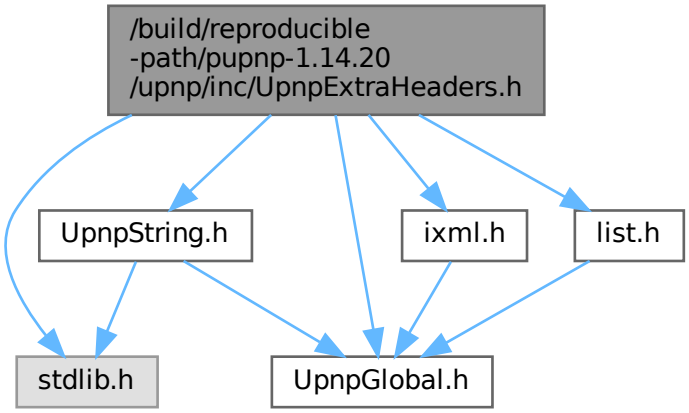
```
00042         const UpnpEventSubscribe *p);
00044 UPNP_EXPORT_SPEC int UpnpEventSubscribe_set_ErrCode(
00045         UpnpEventSubscribe *p, int n);
00046
00048 UPNP_EXPORT_SPEC int UpnpEventSubscribe_get_TimeOut(
00049         const UpnpEventSubscribe *p);
00051 UPNP_EXPORT_SPEC int UpnpEventSubscribe_set_TimeOut(
00052         UpnpEventSubscribe *p, int n);
00053
00055 UPNP_EXPORT_SPEC const UpnpString *UpnpEventSubscribe_get_SID(
00056         const UpnpEventSubscribe *p);
00058 UPNP_EXPORT_SPEC int UpnpEventSubscribe_set_SID(
00059         UpnpEventSubscribe *p, const UpnpString *s);
00061 UPNP_EXPORT_SPEC size_t UpnpEventSubscribe_get_SID_Length(
00062         const UpnpEventSubscribe *p);
00064 UPNP_EXPORT_SPEC const char *UpnpEventSubscribe_get_SID_cstr(
00065         const UpnpEventSubscribe *p);
00067 UPNP_EXPORT_SPEC int UpnpEventSubscribe_strcpy_SID(
00068         UpnpEventSubscribe *p, const char *s);
00070 UPNP_EXPORT_SPEC int UpnpEventSubscribe_strncpy_SID(
00071         UpnpEventSubscribe *p, const char *s, size_t n);
00073 UPNP_EXPORT_SPEC void UpnpEventSubscribe_clear_SID(UpnpEventSubscribe *p);
00074
00076 UPNP_EXPORT_SPEC const UpnpString *UpnpEventSubscribe_get_PublisherUrl(
00077         const UpnpEventSubscribe *p);
00079 UPNP_EXPORT_SPEC int UpnpEventSubscribe_set_PublisherUrl(
00080         UpnpEventSubscribe *p, const UpnpString *s);
00082 UPNP_EXPORT_SPEC size_t UpnpEventSubscribe_get_PublisherUrl_Length(
00083         const UpnpEventSubscribe *p);
00085 UPNP_EXPORT_SPEC const char *UpnpEventSubscribe_get_PublisherUrl_cstr(
00086         const UpnpEventSubscribe *p);
00088 UPNP_EXPORT_SPEC int UpnpEventSubscribe_strcpy_PublisherUrl(
00089         UpnpEventSubscribe *p, const char *s);
00091 UPNP_EXPORT_SPEC int UpnpEventSubscribe_strncpy_PublisherUrl(
00092         UpnpEventSubscribe *p, const char *s, size_t n);
00094 UPNP_EXPORT_SPEC void UpnpEventSubscribe_clear_PublisherUrl(
00095         UpnpEventSubscribe *p);
00096
00097 #ifdef __cplusplus
00098 }
00099 #endif /* __cplusplus */
00100
00101 #endif /* UPNPEVENTSUBSCRIBE_H */
```

9.43 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpExtra↩ Headers.h File Reference

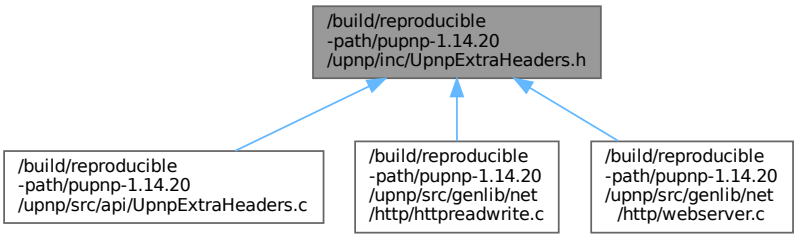
Header file for UpnpExtraHeaders methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpString.h"
#include "ixml.h"
#include "list.h"
```

Include dependency graph for UpnpExtraHeaders.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpExtraHeaders UpnpExtraHeaders

Functions

- UpnpExtraHeaders * UpnpExtraHeaders_new ()
- void UpnpExtraHeaders_delete (UpnpExtraHeaders *p)
- UpnpExtraHeaders * UpnpExtraHeaders_dup (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_assign (UpnpExtraHeaders *p, const UpnpExtraHeaders *q)
- const UpnpListHead * UpnpExtraHeaders_get_node (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_node (UpnpExtraHeaders *p, const UpnpListHead *q)
- void UpnpExtraHeaders_add_to_list_node (UpnpExtraHeaders *p, UpnpListHead *head)
- const UpnpString * UpnpExtraHeaders_get_name (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_name (UpnpExtraHeaders *p, const UpnpString *s)

- size_t UpnpExtraHeaders_get_name_Length (const UpnpExtraHeaders *p)
- const char * UpnpExtraHeaders_get_name_cstr (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_strcpy_name (UpnpExtraHeaders *p, const char *s)
- int UpnpExtraHeaders_strncpy_name (UpnpExtraHeaders *p, const char *s, size_t n)
- void UpnpExtraHeaders_clear_name (UpnpExtraHeaders *p)
- const UpnpString * UpnpExtraHeaders_get_value (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_value (UpnpExtraHeaders *p, const UpnpString *s)
- size_t UpnpExtraHeaders_get_value_Length (const UpnpExtraHeaders *p)
- const char * UpnpExtraHeaders_get_value_cstr (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_strcpy_value (UpnpExtraHeaders *p, const char *s)
- int UpnpExtraHeaders_strncpy_value (UpnpExtraHeaders *p, const char *s, size_t n)
- void UpnpExtraHeaders_clear_value (UpnpExtraHeaders *p)
- const DOMString UpnpExtraHeaders_get_resp (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_resp (UpnpExtraHeaders *p, const DOMString s)
- const char * UpnpExtraHeaders_get_resp_cstr (const UpnpExtraHeaders *p)

9.43.1 Detailed Description

Header file for UpnpExtraHeaders methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.43.2 Typedef Documentation

9.43.2.1 UpnpExtraHeaders

```
typedef struct s_UpnpExtraHeaders UpnpExtraHeaders
```

UpnpExtraHeaders

9.43.3 Function Documentation

9.43.3.1 UpnpExtraHeaders_add_to_list_node()

```
void UpnpExtraHeaders_add_to_list_node (
            UpnpExtraHeaders * p,
            UpnpListHead * head )
```

UpnpExtraHeaders_add_to_list_node

9.43.3.2 UpnpExtraHeaders_assign()

```
int UpnpExtraHeaders_assign (
            UpnpExtraHeaders * p,
            const UpnpExtraHeaders * q )
```

Assignment operator

References UpnpExtraHeaders_get_name(), UpnpExtraHeaders_get_node(), UpnpExtraHeaders_get_resp(), UpnpExtraHeaders_get_value(), UpnpExtraHeaders_set_name(), UpnpExtraHeaders_set_node(), UpnpExtraHeaders_set_resp(), and UpnpExtraHeaders_set_value().

Referenced by UpnpExtraHeaders_dup().

9.43.3.3 UpnpExtraHeaders_clear_name()

```
void UpnpExtraHeaders_clear_name (
            UpnpExtraHeaders * p )
```

UpnpExtraHeaders_clear_name

References UpnpString_clear().

9.43.3.4 UpnpExtraHeaders_clear_value()

```
void UpnpExtraHeaders_clear_value (
            UpnpExtraHeaders * p )
```

UpnpExtraHeaders_clear_value

References UpnpString_clear().

9.43.3.5 UpnpExtraHeaders_delete()

```
void UpnpExtraHeaders_delete (
            UpnpExtraHeaders * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by FreeExtraHTTPHeaders().

9.43.3.6 UpnpExtraHeaders_dup()

```
UpnpExtraHeaders * UpnpExtraHeaders_dup (
            const UpnpExtraHeaders * p )
```

Copy Constructor

References UpnpExtraHeaders_assign(), and UpnpExtraHeaders_new().

9.43.3.7 UpnpExtraHeaders_get_name()

```
const UpnpString * UpnpExtraHeaders_get_name (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_name

Referenced by UpnpExtraHeaders_assign(), UpnpExtraHeaders_get_name_cstr(), and UpnpExtraHeaders_get_name_Length().

9.43.3.8 UpnpExtraHeaders_get_name_cstr()

```
const char * UpnpExtraHeaders_get_name_cstr (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_name_cstr

References UpnpExtraHeaders_get_name(), and UpnpString_get_String().

9.43.3.9 UpnpExtraHeaders_get_name_Length()

```
size_t UpnpExtraHeaders_get_name_Length (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_name_Length

References UpnpExtraHeaders_get_name(), and UpnpString_get_Length().

9.43.3.10 UpnpExtraHeaders_get_node()

```
const UpnpListHead * UpnpExtraHeaders_get_node (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_node

Referenced by ExtraHTTPHeaders(), and UpnpExtraHeaders_assign().

9.43.3.11 UpnpExtraHeaders_get_resp()

```
const DOMString UpnpExtraHeaders_get_resp (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_resp

Referenced by http_MakeMessage(), UpnpExtraHeaders_assign(), and UpnpExtraHeaders_get_resp_cstr().

9.43.3.12 UpnpExtraHeaders_get_resp_cstr()

```
const char * UpnpExtraHeaders_get_resp_cstr (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_resp_cstr

References UpnpExtraHeaders_get_resp().

9.43.3.13 UpnpExtraHeaders_get_value()

```
const UpnpString * UpnpExtraHeaders_get_value (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_value

Referenced by UpnpExtraHeaders_assign(), UpnpExtraHeaders_get_value_cstr(), and UpnpExtraHeaders_get_value_Length().

9.43.3.14 UpnpExtraHeaders_get_value_cstr()

```
const char * UpnpExtraHeaders_get_value_cstr (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_value_cstr

References UpnpExtraHeaders_get_value(), and UpnpString_get_String().

9.43.3.15 UpnpExtraHeaders_get_value_Length()

```
size_t UpnpExtraHeaders_get_value_Length (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_value_Length

References UpnpExtraHeaders_get_value(), and UpnpString_get_Length().

9.43.3.16 UpnpExtraHeaders_new()

```
UpnpExtraHeaders * UpnpExtraHeaders_new ( )
```

Constructor

References UpnpString_new().

Referenced by ExtraHTTPHeaders(), and UpnpExtraHeaders_dup().

9.43.3.17 UpnpExtraHeaders_set_name()

```
int UpnpExtraHeaders_set_name (
            UpnpExtraHeaders * p,
            const UpnpString * s )
```

UpnpExtraHeaders_set_name

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpExtraHeaders_assign().

9.43.3.18 UpnpExtraHeaders_set_node()

```
int UpnpExtraHeaders_set_node (
            UpnpExtraHeaders * p,
            const UpnpListHead * q )
```

UpnpExtraHeaders_set_node

Referenced by UpnpExtraHeaders_assign().

9.43.3.19 UpnpExtraHeaders_set_resp()

```
int UpnpExtraHeaders_set_resp (
            UpnpExtraHeaders * p,
            const DOMString s )
```

UpnpExtraHeaders_set_resp

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by UpnpExtraHeaders_assign().

9.43.3.20 UpnpExtraHeaders_set_value()

```
int UpnpExtraHeaders_set_value (
            UpnpExtraHeaders * p,
            const UpnpString * s )
```

UpnpExtraHeaders_set_value

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpExtraHeaders_assign().

9.43.3.21 UpnpExtraHeaders_strcpy_name()

```
int UpnpExtraHeaders_strcpy_name (
            UpnpExtraHeaders * p,
            const char * s )
```

UpnpExtraHeaders_strcpy_name

References UpnpString_set_String().

9.43.3.22 UpnpExtraHeaders_strcpy_value()

```
int UpnpExtraHeaders_strcpy_value (
            UpnpExtraHeaders * p,
            const char * s )
```

UpnpExtraHeaders_strcpy_value

References UpnpString_set_String().

9.43.3.23 UpnpExtraHeaders_strncpy_name()

```
int UpnpExtraHeaders_strncpy_name (
             UpnpExtraHeaders * p,
             const char * s,
             size_t n )
```

UpnpExtraHeaders_strncpy_name

References UpnpString_set_StringN().

Referenced by ExtraHTTPHeaders().

9.43.3.24 UpnpExtraHeaders_strncpy_value()

```
int UpnpExtraHeaders_strncpy_value (
             UpnpExtraHeaders * p,
             const char * s,
             size_t n )
```

UpnpExtraHeaders_strncpy_value

References UpnpString_set_StringN().

Referenced by ExtraHTTPHeaders().

9.44 UpnpExtraHeaders.h

Go to the documentation of this file.

```
00001 #ifndef UPNPEXTRAHEADERS_H
00002 #define UPNPEXTRAHEADERS_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpString.h"
00019 #include "ixml.h"
00020 #include "list.h"
00021
00022 #ifdef __cplusplus
00023 extern "C" {
00024 #endif /* __cplusplus */
00025
00029 typedef struct s_UpnpExtraHeaders UpnpExtraHeaders;
00030
00032 UPNP_EXPORT_SPEC UpnpExtraHeaders *UpnpExtraHeaders_new();
00034 UPNP_EXPORT_SPEC void UpnpExtraHeaders_delete(UpnpExtraHeaders *p);
00036 UPNP_EXPORT_SPEC UpnpExtraHeaders *UpnpExtraHeaders_dup(
00037         const UpnpExtraHeaders *p);
00039 UPNP_EXPORT_SPEC int UpnpExtraHeaders_assign(
00040         UpnpExtraHeaders *p, const UpnpExtraHeaders *q);
00041
00043 UPNP_EXPORT_SPEC const UpnpListHead *UpnpExtraHeaders_get_node(
00044         const UpnpExtraHeaders *p);
00046 UPNP_EXPORT_SPEC int UpnpExtraHeaders_set_node(
00047         UpnpExtraHeaders *p, const UpnpListHead *q);
00049 UPNP_EXPORT_SPEC void UpnpExtraHeaders_add_to_list_node(
00050         UpnpExtraHeaders *p, UpnpListHead *head);
00051
00053 UPNP_EXPORT_SPEC const UpnpString *UpnpExtraHeaders_get_name(
00054         const UpnpExtraHeaders *p);
00056 UPNP_EXPORT_SPEC int UpnpExtraHeaders_set_name(
00057         UpnpExtraHeaders *p, const UpnpString *s);
00059 UPNP_EXPORT_SPEC size_t UpnpExtraHeaders_get_name_Length(
00060         const UpnpExtraHeaders *p);
```

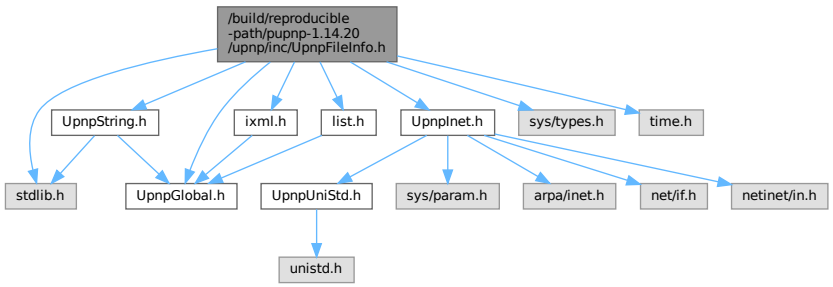
```
00062 UPNP_EXPORT_SPEC const char *UpnpExtraHeaders_get_name_cstr(
00063         const UpnpExtraHeaders *p);
00065 UPNP_EXPORT_SPEC int UpnpExtraHeaders_strcpy_name(
00066         UpnpExtraHeaders *p, const char *s);
00068 UPNP_EXPORT_SPEC int UpnpExtraHeaders_strncpy_name(
00069         UpnpExtraHeaders *p, const char *s, size_t n);
00071 UPNP_EXPORT_SPEC void UpnpExtraHeaders_clear_name(UpnpExtraHeaders *p);
00072
00074 UPNP_EXPORT_SPEC const UpnpString *UpnpExtraHeaders_get_value(
00075         const UpnpExtraHeaders *p);
00077 UPNP_EXPORT_SPEC int UpnpExtraHeaders_set_value(
00078         UpnpExtraHeaders *p, const UpnpString *s);
00080 UPNP_EXPORT_SPEC size_t UpnpExtraHeaders_get_value_Length(
00081         const UpnpExtraHeaders *p);
00083 UPNP_EXPORT_SPEC const char *UpnpExtraHeaders_get_value_cstr(
00084         const UpnpExtraHeaders *p);
00086 UPNP_EXPORT_SPEC int UpnpExtraHeaders_strcpy_value(
00087         UpnpExtraHeaders *p, const char *s);
00089 UPNP_EXPORT_SPEC int UpnpExtraHeaders_strncpy_value(
00090         UpnpExtraHeaders *p, const char *s, size_t n);
00092 UPNP_EXPORT_SPEC void UpnpExtraHeaders_clear_value(UpnpExtraHeaders *p);
00093
00095 UPNP_EXPORT_SPEC const DOMString UpnpExtraHeaders_get_resp(
00096         const UpnpExtraHeaders *p);
00098 UPNP_EXPORT_SPEC int UpnpExtraHeaders_set_resp(
00099         UpnpExtraHeaders *p, const DOMString s);
00101 UPNP_EXPORT_SPEC const char *UpnpExtraHeaders_get_resp_cstr(
00102         const UpnpExtraHeaders *p);
00103
00104 #ifdef __cplusplus
00105 }
00106 #endif /* __cplusplus */
00107
00108 #endif /* UPNPEXTRAHEADERS_H */
```

9.45 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpFileInfo.h File Reference

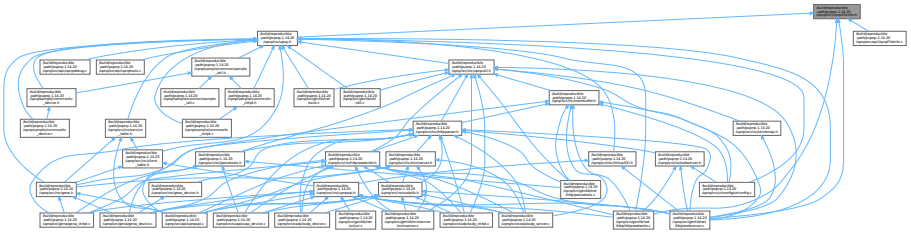
Header file for UpnpFileInfo methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include "UpnpString.h"
#include "ixml.h"
#include "list.h"
#include <sys/types.h>
#include <time.h>
```

Include dependency graph for UpnpFileInfo.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpFileInfo UpnpFileInfo

Functions

- UpnpFileInfo * UpnpFileInfo_new ()
- void UpnpFileInfo_delete (UpnpFileInfo *p)
- UpnpFileInfo * UpnpFileInfo_dup (const UpnpFileInfo *p)
- int UpnpFileInfo_assign (UpnpFileInfo *p, const UpnpFileInfo *q)
- off_t UpnpFileInfo_get_FileLength (const UpnpFileInfo *p)
- int UpnpFileInfo_set_FileLength (UpnpFileInfo *p, off_t n)
- time_t UpnpFileInfo_get_LastModified (const UpnpFileInfo *p)
- int UpnpFileInfo_set_LastModified (UpnpFileInfo *p, time_t n)
- int UpnpFileInfo_get_IsDirectory (const UpnpFileInfo *p)
- int UpnpFileInfo_set_IsDirectory (UpnpFileInfo *p, int n)
- int UpnpFileInfo_get_IsReadable (const UpnpFileInfo *p)
- int UpnpFileInfo_set_IsReadable (UpnpFileInfo *p, int n)
- const DOMString UpnpFileInfo_get_ContentType (const UpnpFileInfo *p)
- int UpnpFileInfo_set_ContentType (UpnpFileInfo *p, const DOMString s)
- const char * UpnpFileInfo_get_ContentType_cstr (const UpnpFileInfo *p)
- const UpnpListHead * UpnpFileInfo_get_ExtraHeadersList (const UpnpFileInfo *p)
- int UpnpFileInfo_set_ExtraHeadersList (UpnpFileInfo *p, const UpnpListHead *q)
- void UpnpFileInfo_add_to_list_ExtraHeadersList (UpnpFileInfo *p, UpnpListHead *head)
- const struct sockaddr_storage * UpnpFileInfo_get_CtrlPtIPAddr (const UpnpFileInfo *p)
- int UpnpFileInfo_set_CtrlPtIPAddr (UpnpFileInfo *p, const struct sockaddr_storage *buf)
- void UpnpFileInfo_clear_CtrlPtIPAddr (UpnpFileInfo *p)
- const UpnpString * UpnpFileInfo_get_Os (const UpnpFileInfo *p)
- int UpnpFileInfo_set_Os (UpnpFileInfo *p, const UpnpString *s)
- size_t UpnpFileInfo_get_Os_Length (const UpnpFileInfo *p)
- const char * UpnpFileInfo_get_Os_cstr (const UpnpFileInfo *p)
- int UpnpFileInfo_strcpy_Os (UpnpFileInfo *p, const char *s)
- int UpnpFileInfo_strncpy_Os (UpnpFileInfo *p, const char *s, size_t n)
- void UpnpFileInfo_clear_Os (UpnpFileInfo *p)

9.45.1 Detailed Description

Header file for UpnpFileInfo methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.45.2 Typedef Documentation

9.45.2.1 UpnpFileInfo

```
typedef struct s_UpnpFileInfo UpnpFileInfo
```

UpnpFileInfo

9.45.3 Function Documentation

9.45.3.1 UpnpFileInfo_add_to_list_ExtraHeadersList()

```
void UpnpFileInfo_add_to_list_ExtraHeadersList (
            UpnpFileInfo * p,
            UpnpListHead * head )
```

UpnpFileInfo_add_to_list_ExtraHeadersList

9.45.3.2 UpnpFileInfo_assign()

```
int UpnpFileInfo_assign (
            UpnpFileInfo * p,
            const UpnpFileInfo * q )
```

Assignment operator

References UpnpFileInfo_get_ContentType(), UpnpFileInfo_get_CtrlPtIPAddr(), UpnpFileInfo_get_ExtraHeadersList(), UpnpFileInfo_get_FileLength(), UpnpFileInfo_get_IsDirectory(), UpnpFileInfo_get_IsReadable(), UpnpFileInfo_get_LastModified(), UpnpFileInfo_get_Os(), UpnpFileInfo_set_ContentType(), UpnpFileInfo_set_CtrlPtIPAddr(), UpnpFileInfo_set_ExtraHeadersList(), UpnpFileInfo_set_FileLength(), UpnpFileInfo_set_IsDirectory(), UpnpFileInfo_set_IsReadable(), UpnpFileInfo_set_LastModified(), and UpnpFileInfo_set_Os().

Referenced by UpnpFileInfo_dup().

9.45.3.3 UpnpFileInfo_clear_CtrlPtIPAddr()

```
void UpnpFileInfo_clear_CtrlPtIPAddr (
            UpnpFileInfo * p )
```

UpnpFileInfo_get_CtrlPtIPAddr

9.45.3.4 UpnpFileInfo_clear_Os()

```
void UpnpFileInfo_clear_Os (
            UpnpFileInfo * p )
```

UpnpFileInfo_clear_Os

References UpnpString_clear().

9.45.3.5 UpnpFileInfo_delete()

```
void UpnpFileInfo_delete (
            UpnpFileInfo * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by process_request().

9.45.3.6 UpnpFileInfo_dup()

```
UpnpFileInfo * UpnpFileInfo_dup (
            const UpnpFileInfo * p )
```

Copy Constructor

References UpnpFileInfo_assign(), and UpnpFileInfo_new().

9.45.3.7 UpnpFileInfo_get_ContentType()

```
const DOMString UpnpFileInfo_get_ContentType (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_ContentType

Referenced by get_content_type(), process_request(), UpnpFileInfo_assign(), and UpnpFileInfo_get_ContentType_cstr().

9.45.3.8 UpnpFileInfo_get_ContentType_cstr()

```
const char * UpnpFileInfo_get_ContentType_cstr (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_ContentType_cstr

References UpnpFileInfo_get_ContentType().

9.45.3.9 UpnpFileInfo_get_CtrlPtIPAddr()

```
const struct sockaddr_storage * UpnpFileInfo_get_CtrlPtIPAddr (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_CtrlPtIPAddr

Referenced by UpnpFileInfo_assign().

9.45.3.10 UpnpFileInfo_get_ExtraHeadersList()

```
const UpnpListHead * UpnpFileInfo_get_ExtraHeadersList (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_ExtraHeadersList

Referenced by process_request(), and UpnpFileInfo_assign().

9.45.3.11 UpnpFileInfo_get_FileLength()

```
off_t UpnpFileInfo_get_FileLength (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_FileLength

Referenced by get_file_info(), process_request(), and UpnpFileInfo_assign().

9.45.3.12 UpnpFileInfo_get_IsDirectory()

```
int UpnpFileInfo_get_IsDirectory (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_IsDirectory

Referenced by process_request(), and UpnpFileInfo_assign().

9.45.3.13 UpnpFileInfo_get_IsReadable()

```
int UpnpFileInfo_get_IsReadable (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_IsReadable

Referenced by get_file_info(), process_request(), and UpnpFileInfo_assign().

9.45.3.14 UpnpFileInfo_get_LastModified()

```
time_t UpnpFileInfo_get_LastModified (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_LastModified

Referenced by get_file_info(), process_request(), and UpnpFileInfo_assign().

9.45.3.15 UpnpFileInfo_get_Os()

```
const UpnpString * UpnpFileInfo_get_Os (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_Os

Referenced by UpnpFileInfo_assign(), UpnpFileInfo_get_Os_cstr(), and UpnpFileInfo_get_Os_Length().

9.45.3.16 UpnpFileInfo_get_Os_cstr()

```
const char * UpnpFileInfo_get_Os_cstr (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_Os_cstr

References UpnpFileInfo_get_Os(), and UpnpString_get_String().

9.45.3.17 UpnpFileInfo_get_Os_Length()

```
size_t UpnpFileInfo_get_Os_Length (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_Os_Length

References UpnpFileInfo_get_Os(), and UpnpString_get_Length().

9.45.3.18 UpnpFileInfo_new()

```
UpnpFileInfo * UpnpFileInfo_new ( )
```

Constructor

References UpnpString_new().

Referenced by process_request(), and UpnpFileInfo_dup().

9.45.3.19 UpnpFileInfo_set_ContentType()

```
int UpnpFileInfo_set_ContentType (
            UpnpFileInfo * p,
            const DOMString s )
```

UpnpFileInfo_set_ContentType

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by get_content_type(), get_file_info(), process_request(), and UpnpFileInfo_assign().

9.45.3.20 UpnpFileInfo_set_CtrlPtIPAddr()

```
int UpnpFileInfo_set_CtrlPtIPAddr (
            UpnpFileInfo * p,
            const struct sockaddr_storage * buf )
```

UpnpFileInfo_get_CtrlPtIPAddr

Referenced by process_request(), and UpnpFileInfo_assign().

9.45.3.21 UpnpFileInfo_set_ExtraHeadersList()

```
int UpnpFileInfo_set_ExtraHeadersList (
            UpnpFileInfo * p,
            const UpnpListHead * q )
```

UpnpFileInfo_set_ExtraHeadersList

Referenced by UpnpFileInfo_assign().

9.45.3.22 UpnpFileInfo_set_FileLength()

```
int UpnpFileInfo_set_FileLength (
            UpnpFileInfo * p,
            off_t n )
```

UpnpFileInfo_set_FileLength

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.45.3.23 UpnpFileInfo_set_IsDirectory()

```
int UpnpFileInfo_set_IsDirectory (
            UpnpFileInfo * p,
            int n )
```

UpnpFileInfo_set_IsDirectory

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.45.3.24 UpnpFileInfo_set_IsReadable()

```
int UpnpFileInfo_set_IsReadable (
            UpnpFileInfo * p,
            int n )
```

UpnpFileInfo_set_IsReadable

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.45.3.25 UpnpFileInfo_set_LastModified()

```
int UpnpFileInfo_set_LastModified (
            UpnpFileInfo * p,
            time_t n )
```

UpnpFileInfo_set_LastModified

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.45.3.26 UpnpFileInfo_set_Os()

```
int UpnpFileInfo_set_Os (
             UpnpFileInfo * p,
             const UpnpString * s )
```

UpnpFileInfo_set_Os

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpFileInfo_assign().

9.45.3.27 UpnpFileInfo_strcpy_Os()

```
int UpnpFileInfo_strcpy_Os (
             UpnpFileInfo * p,
             const char * s )
```

UpnpFileInfo_strcpy_Os

References UpnpString_set_String().

9.45.3.28 UpnpFileInfo_strncpy_Os()

```
int UpnpFileInfo_strncpy_Os (
             UpnpFileInfo * p,
             const char * s,
             size_t n )
```

UpnpFileInfo_strncpy_Os

References UpnpString_set_StringN().

Referenced by process_request().

9.46 UpnpFileInfo.h

Go to the documentation of this file.

```
00001 #ifndef UPNPFILEINFO_H
00002 #define UPNPFILEINFO_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpInet.h"
00019 #include "UpnpString.h"
00020 #include "ixml.h"
00021 #include "list.h"
00022 #include <sys/types.h>
00023 #include <time.h>
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif /* __cplusplus */
00028
00032 typedef struct s_UpnpFileInfo UpnpFileInfo;
00033
00035 UPNP_EXPORT_SPEC UpnpFileInfo *UpnpFileInfo_new();
```

```
00037 UPNP_EXPORT_SPEC void UpnpFileInfo_delete(UpnpFileInfo *p);
00039 UPNP_EXPORT_SPEC UpnpFileInfo *UpnpFileInfo_dup(const UpnpFileInfo *p);
00041 UPNP_EXPORT_SPEC int UpnpFileInfo_assign(
00042         UpnpFileInfo *p, const UpnpFileInfo *q);
00043
00045 UPNP_EXPORT_SPEC off_t UpnpFileInfo_get_FileLength(const UpnpFileInfo *p);
00047 UPNP_EXPORT_SPEC int UpnpFileInfo_set_FileLength(UpnpFileInfo *p, off_t n);
00048
00050 UPNP_EXPORT_SPEC time_t UpnpFileInfo_get_LastModified(const UpnpFileInfo *p);
00052 UPNP_EXPORT_SPEC int UpnpFileInfo_set_LastModified(UpnpFileInfo *p, time_t n);
00053
00055 UPNP_EXPORT_SPEC int UpnpFileInfo_get_IsDirectory(const UpnpFileInfo *p);
00057 UPNP_EXPORT_SPEC int UpnpFileInfo_set_IsDirectory(UpnpFileInfo *p, int n);
00058
00060 UPNP_EXPORT_SPEC int UpnpFileInfo_get_IsReadable(const UpnpFileInfo *p);
00062 UPNP_EXPORT_SPEC int UpnpFileInfo_set_IsReadable(UpnpFileInfo *p, int n);
00063
00065 UPNP_EXPORT_SPEC const DOMString UpnpFileInfo_get_ContentType(
00066         const UpnpFileInfo *p);
00068 UPNP_EXPORT_SPEC int UpnpFileInfo_set_ContentType(
00069         UpnpFileInfo *p, const DOMString s);
00071 UPNP_EXPORT_SPEC const char *UpnpFileInfo_get_ContentType_cstr(
00072         const UpnpFileInfo *p);
00073
00075 UPNP_EXPORT_SPEC const UpnpListHead *UpnpFileInfo_get_ExtraHeadersList(
00076         const UpnpFileInfo *p);
00078 UPNP_EXPORT_SPEC int UpnpFileInfo_set_ExtraHeadersList(
00079         UpnpFileInfo *p, const UpnpListHead *q);
00081 UPNP_EXPORT_SPEC void UpnpFileInfo_add_to_list_ExtraHeadersList(
00082         UpnpFileInfo *p, UpnpListHead *head);
00083
00085 UPNP_EXPORT_SPEC const struct sockaddr_storage *UpnpFileInfo_get_CtrlPtIPAddr(
00086         const UpnpFileInfo *p);
00088 UPNP_EXPORT_SPEC int UpnpFileInfo_set_CtrlPtIPAddr(
00089         UpnpFileInfo *p, const struct sockaddr_storage *buf);
00091 UPNP_EXPORT_SPEC void UpnpFileInfo_clear_CtrlPtIPAddr(UpnpFileInfo *p);
00092
00094 UPNP_EXPORT_SPEC const UpnpString *UpnpFileInfo_get_Os(const UpnpFileInfo *p);
00096 UPNP_EXPORT_SPEC int UpnpFileInfo_set_Os(UpnpFileInfo *p, const UpnpString *s);
00098 UPNP_EXPORT_SPEC size_t UpnpFileInfo_get_Os_Length(const UpnpFileInfo *p);
00100 UPNP_EXPORT_SPEC const char *UpnpFileInfo_get_Os_cstr(const UpnpFileInfo *p);
00102 UPNP_EXPORT_SPEC int UpnpFileInfo_strcpy_Os(UpnpFileInfo *p, const char *s);
00104 UPNP_EXPORT_SPEC int UpnpFileInfo_strncpy_Os(
00105         UpnpFileInfo *p, const char *s, size_t n);
00107 UPNP_EXPORT_SPEC void UpnpFileInfo_clear_Os(UpnpFileInfo *p);
00108
00109 #ifdef __cplusplus
00110 }
00111 #endif /* __cplusplus */
00112
00113 #endif /* UPNPFILEINFO_H */
```

9.47 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpGlobal.h File Reference

Defines constants that for some reason are not defined on some systems.

This graph shows which files directly or indirectly include this file:

Macros

- #define UPNP_EXPORT_SPEC

  Export functions on WIN32 DLLs.
- #define UPNP_INLINE inline

Declares an inline function.

- #define PRIzd "zd"

  Supply the PRId64 printf() macro.
- #define **PRIzu** "zu"
- #define **PRIzx** "zx"

9.47.1 Detailed Description

Defines constants that for some reason are not defined on some systems.

9.47.2 Macro Definition Documentation

9.47.2.1 PRIzd

```
#define PRIzd "zd"
```

Supply the PRId64 printf() macro.

MSVC still does not know about this.

Supply the PRIz∗ printf() macros.

These macros were invented so that we can live a little longer with MSVC lack of C99. "z" is the correct printf() size specifier for the size_t type.

9.47.2.2 UPNP_EXPORT_SPEC

```
#define UPNP_EXPORT_SPEC
```

Export functions on WIN32 DLLs.

Every funtion that belongs to the library API must use this definition upon declaration or it will not be exported

9.47.2.3 UPNP_INLINE

```
#define UPNP_INLINE inline
```

Declares an inline function.

Surprisingly, there are some compilers that do not understand the inline keyword. This definition makes the use of this keyword portable to these systems.

9.48 UpnpGlobal.h

Go to the documentation of this file.

```
00001 #ifndef UPNPGLOBAL_H
00002 #define UPNPGLOBAL_H
00003
00011 #if defined UPNP_LARGEFILE_SENSITIVE && _FILE_OFFSET_BITS + 0 != 64
00012         #if defined __GNUC__
00013                 #warning libupnp requires largefile mode - use AC_SYS_LARGEFILE
00014         #elif !defined _WIN32
00015                 #error libupnp requires largefile mode - use AC_SYS_LARGEFILE
00016         #endif
00017 #endif
00018
00019 #ifdef _WIN32
00020         /*
00021          * UPNP_EXPORT_SPEC
00022          */
00023         #if defined _MSC_VER || defined __BORLANDC__
00024                 #ifdef UPNP_STATIC_LIB
00025                         #define UPNP_EXPORT_SPEC
00026                 #else /* UPNP_STATIC_LIB */
00027                         #ifdef LIBUPNP_EXPORTS
00030                                 #define UPNP_EXPORT_SPEC __declspec(dllexport)
00031                         #else /* LIBUPNP_EXPORTS */
00032                                 #define UPNP_EXPORT_SPEC __declspec(dllimport)
00033                         #endif /* LIBUPNP_EXPORTS */
00034                 #endif         /* UPNP_STATIC_LIB */
00035         #else                  /* _MSC_VER || __BORLANDC__ */
00036                 #define UPNP_EXPORT_SPEC
00037         #endif /* _MSC_VER || __BORLANDC__ */
00038
00039         /*
00040          * UPNP_INLINE
00041          * PRId64
00042          * PRIzd
00043          * PRIzu
00044          * PRIzx
00045          */
00046         #ifdef UPNP_USE_MSVCPP
00047                 #if _MSC_VER > 1900
00048                         #define UPNP_INLINE inline
00049                         #define PRIzd "zd"
00050                         #define PRIzu "zu"
00051                         #define PRIzx "zx"
00052                 #else
00053                         /* define some things the M$ VC++ doesn't know */
00054                         #define UPNP_INLINE _inline
00055 typedef __int64 int64_t;
00056                         #define PRIzd "ld"
00057                         #define PRIzu "lu"
00058                         #define PRIzx "lx"
00059                 #endif
00060         #endif /* UPNP_USE_MSVCPP */
00061
00062         #ifdef UPNP_USE_BCBPP
00063                 /* define some things Borland Builder doesn't know */
00064                 /* inconsistency between the httpparser.h and the .c file
00065                    definition. Header is missing UPNP_INLINE prefix, so compiler
00066                    is confused ... better remove it #define UPNP_INLINE inline
00067                  */
00068                 #define UPNP_INLINE
00069 typedef __int64 int64_t;
00070                 #warning The Borland C compiler is probably broken on PRId64,
00071                 #warning please someone provide a proper fix here
00072                 #define PRId64 "Ld"
00073                 #define PRIzd "ld"
00074                 #define PRIzu "lu"
00075                 #define PRIzx "lx"
00076                 #define SCNd64 "Ld"
00077         #endif /* UPNP_USE_BCBPP */
00078
00079         #ifdef __GNUC__
00080                 #define UPNP_INLINE inline
00081                 /* Note with PRIzu that in the case of Mingw32, it's the MS C
00082                  * runtime printf which ends up getting called, not the glibc
00083                  * printf, so it genuinely doesn't have "zu"
00084                  */
00085                 #define PRIzd "ld"
00086                 #define PRIzu "lu"
00087                 #define PRIzx "lx"
00088         #endif /* __GNUC__ */
00089 #else
00096         #ifdef UPNP_STATIC_LIB
00097                 #define UPNP_EXPORT_SPEC
```

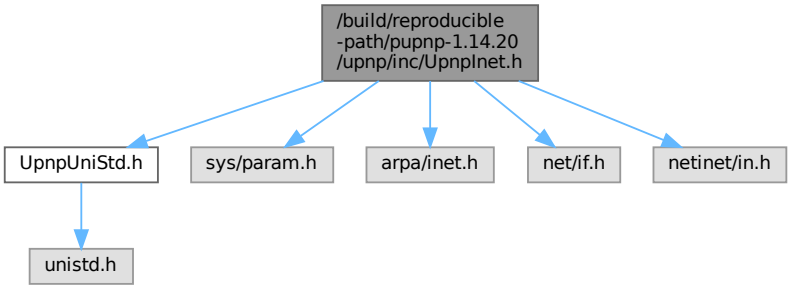
```
00098         #else /* UPNP_STATIC_LIB */
00099                 #ifdef LIBUPNP_EXPORTS
00102                         #define UPNP_EXPORT_SPEC \
00103                                 __attribute__((visibility("default")))
00104                 #else /* LIBUPNP_EXPORTS */
00105                         #define UPNP_EXPORT_SPEC
00106                 #endif /* LIBUPNP_EXPORTS */
00107         #endif
00108
00116         #ifdef __STRICT_ANSI__
00117                 #define UPNP_INLINE __inline__
00118         #else
00119                 #define UPNP_INLINE inline
00120         #endif
00121
00127         /* #define PRId64 PRId64 */
00128
00136         #define PRIzd "zd"
00137         #define PRIzu "zu"
00138         #define PRIzx "zx"
00139 #endif
00140
00141 /*
00142  * Defining this macro here gives some interesting information about unused
00143  * functions in the code. Of course, this should never go uncommented on a
00144  * release.
00145  */
00146 /*#define inline*/
00147
00148 #endif /* UPNPGLOBAL_H */
```

9.49 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpInet.h File Reference

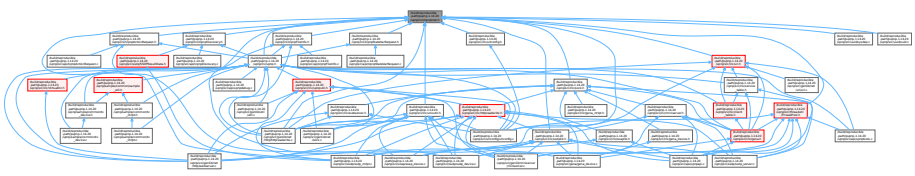
Provides a platform independent way to include TCP/IP types and functions.

```
#include "UpnpUniStd.h"
#include <sys/param.h>
#include <arpa/inet.h>
#include <net/if.h>
#include <netinet/in.h>
```

Include dependency graph for UpnpInet.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define INVALID_SOCKET (-1)
- #define SOCKET_ERROR (-1)
- #define UpnpCloseSocket close

Typedefs

- typedef int SOCKET

9.49.1 Detailed Description

Provides a platform independent way to include TCP/IP types and functions.

9.50 UpnpInet.h

Go to the documentation of this file.

```
00001 #ifndef UPNPINET_H
00002 #define UPNPINET_H
00003
00015 #include "UpnpUniStd.h" /* for close() */
00016
00017 #ifdef _WIN32
00018         #include <stdarg.h>
00019         #include <winsock2.h>
00020         #include <ws2tcpip.h>
00021
00022         #define UpnpCloseSocket closesocket
00023
00024         #if (_WIN32_WINNT < 0x0600)
00025 typedef short sa_family_t;
00026         #else
00027 typedef ADDRESS_FAMILY sa_family_t;
00028         #endif
00029
00030 #else /* _WIN32 */
00031         #include <sys/param.h>
00032         #if defined(__sun)
00033                 #include <fcntl.h>
00034                 #include <sys/sockio.h>
00035         #elif (defined(BSD) && BSD >= 199306) || defined(__FreeBSD_kernel__)
00036                 #include <ifaddrs.h>
00037                 /* Do not move or remove the include below for "sys/socket"!
00038                  * Will break FreeBSD builds. */
00039                 #include <sys/socket.h>
00040         #endif
00041         #include <arpa/inet.h> /* for inet_pton() */
00042         #include <net/if.h>
00043         #include <netinet/in.h>
00044
00048 typedef int SOCKET;
00049
00051         #define INVALID_SOCKET (-1)
00052
00054         #define SOCKET_ERROR (-1)
00055
00057         #define UpnpCloseSocket close
00058 #endif /* _WIN32 */
00059
00060 /* @} Sock */
00061
00062 #endif /* UPNPINET_H */
```

9.51 UpnpIntTypes.h

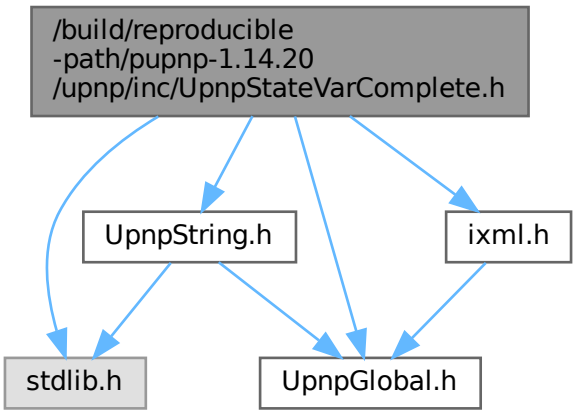
```
00001 #ifndef UPNPINTTYPES_H
00002 #define UPNPINTTYPES_H
00003
00004 #if !defined(UPNP_USE_BCBPP)
00005
00006         /* Printf format for integers. */
00007         #include <inttypes.h>
00008
00009 #endif /* !defined(UPNP_USE_BCBPP) */
00010
00011 #endif /* UPNPINTTYPES_H */
```

9.52 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarComplete.h File Reference

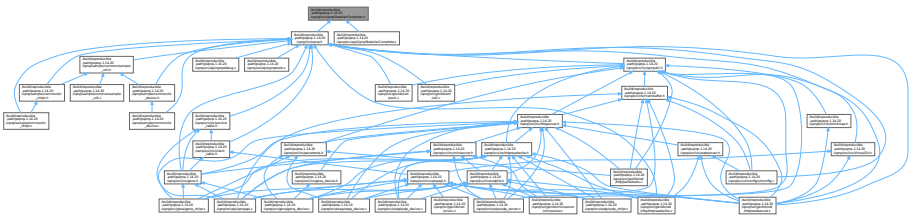
Header file for UpnpStateVarComplete methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpString.h"
#include "ixml.h"
```

Include dependency graph for UpnpStateVarComplete.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpStateVarComplete UpnpStateVarComplete

Functions

- UpnpStateVarComplete * UpnpStateVarComplete_new ()
- void UpnpStateVarComplete_delete (UpnpStateVarComplete *p)
- UpnpStateVarComplete * UpnpStateVarComplete_dup (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_assign (UpnpStateVarComplete *p, const UpnpStateVarComplete *q)
- int UpnpStateVarComplete_get_ErrCode (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_ErrCode (UpnpStateVarComplete *p, int n)
- const UpnpString * UpnpStateVarComplete_get_CtrlUrl (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_CtrlUrl (UpnpStateVarComplete *p, const UpnpString *s)
- size_t UpnpStateVarComplete_get_CtrlUrl_Length (const UpnpStateVarComplete *p)
- const char * UpnpStateVarComplete_get_CtrlUrl_cstr (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_strcpy_CtrlUrl (UpnpStateVarComplete *p, const char *s)
- int UpnpStateVarComplete_strncpy_CtrlUrl (UpnpStateVarComplete *p, const char *s, size_t n)
- void UpnpStateVarComplete_clear_CtrlUrl (UpnpStateVarComplete *p)
- const UpnpString * UpnpStateVarComplete_get_StateVarName (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_StateVarName (UpnpStateVarComplete *p, const UpnpString *s)
- size_t UpnpStateVarComplete_get_StateVarName_Length (const UpnpStateVarComplete *p)
- const char * UpnpStateVarComplete_get_StateVarName_cstr (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_strcpy_StateVarName (UpnpStateVarComplete *p, const char *s)
- int UpnpStateVarComplete_strncpy_StateVarName (UpnpStateVarComplete *p, const char *s, size_t n)
- void UpnpStateVarComplete_clear_StateVarName (UpnpStateVarComplete *p)
- const DOMString UpnpStateVarComplete_get_CurrentVal (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_CurrentVal (UpnpStateVarComplete *p, const DOMString s)
- const char * UpnpStateVarComplete_get_CurrentVal_cstr (const UpnpStateVarComplete *p)

9.52.1 Detailed Description

Header file for UpnpStateVarComplete methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.52.2 Typedef Documentation

9.52.2.1 UpnpStateVarComplete

```
typedef struct s_UpnpStateVarComplete UpnpStateVarComplete
```

UpnpStateVarComplete

9.52.3 Function Documentation

9.52.3.1 UpnpStateVarComplete_assign()

```
int UpnpStateVarComplete_assign (
            UpnpStateVarComplete * p,
            const UpnpStateVarComplete * q )
```

Assignment operator

References UpnpStateVarComplete_get_CtrlUrl(), UpnpStateVarComplete_get_CurrentVal(), UpnpStateVarComplete_get_ErrCode(), UpnpStateVarComplete_get_StateVarName(), UpnpStateVarComplete_set_CtrlUrl(), UpnpStateVarComplete_set_CurrentVal(), UpnpStateVarComplete_set_ErrCode(), and UpnpStateVarComplete_set_StateVarName().

Referenced by UpnpStateVarComplete_dup().

9.52.3.2 UpnpStateVarComplete_clear_CtrlUrl()

```
void UpnpStateVarComplete_clear_CtrlUrl (
            UpnpStateVarComplete * p )
```

UpnpStateVarComplete_clear_CtrlUrl

References UpnpString_clear().

9.52.3.3 UpnpStateVarComplete_clear_StateVarName()

```
void UpnpStateVarComplete_clear_StateVarName (
            UpnpStateVarComplete * p )
```

UpnpStateVarComplete_clear_StateVarName

References UpnpString_clear().

9.52.3.4 UpnpStateVarComplete_delete()

```
void UpnpStateVarComplete_delete (
            UpnpStateVarComplete * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by UpnpThreadDistribution().

9.52.3.5 UpnpStateVarComplete_dup()

```
UpnpStateVarComplete * UpnpStateVarComplete_dup (
            const UpnpStateVarComplete * p )
```

Copy Constructor

References UpnpStateVarComplete_assign(), and UpnpStateVarComplete_new().

9.52.3.6 UpnpStateVarComplete_get_CtrlUrl()

```
const UpnpString * UpnpStateVarComplete_get_CtrlUrl (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CtrlUrl

Referenced by SampleUtil_PrintEvent(), UpnpStateVarComplete_assign(), UpnpStateVarComplete_get_CtrlUrl_cstr(), and UpnpStateVarComplete_get_CtrlUrl_Length().

9.52.3.7 UpnpStateVarComplete_get_CtrlUrl_cstr()

```
const char * UpnpStateVarComplete_get_CtrlUrl_cstr (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CtrlUrl_cstr

References UpnpStateVarComplete_get_CtrlUrl(), and UpnpString_get_String().

9.52.3.8 UpnpStateVarComplete_get_CtrlUrl_Length()

```
size_t UpnpStateVarComplete_get_CtrlUrl_Length (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CtrlUrl_Length

References UpnpStateVarComplete_get_CtrlUrl(), and UpnpString_get_Length().

9.52.3.9 UpnpStateVarComplete_get_CurrentVal()

```
const DOMString UpnpStateVarComplete_get_CurrentVal (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CurrentVal

Referenced by SampleUtil_PrintEvent(), UpnpStateVarComplete_assign(), and UpnpStateVarComplete_get_CurrentVal_cstr().

9.52.3.10 UpnpStateVarComplete_get_CurrentVal_cstr()

```
const char * UpnpStateVarComplete_get_CurrentVal_cstr (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CurrentVal_cstr

References UpnpStateVarComplete_get_CurrentVal().

9.52.3.11 UpnpStateVarComplete_get_ErrCode()

```
int UpnpStateVarComplete_get_ErrCode (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpStateVarComplete_assign().

9.52.3.12 UpnpStateVarComplete_get_StateVarName()

```
const UpnpString * UpnpStateVarComplete_get_StateVarName (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_StateVarName

Referenced by SampleUtil_PrintEvent(), UpnpStateVarComplete_assign(), UpnpStateVarComplete_get_StateVarName_cstr(), and UpnpStateVarComplete_get_StateVarName_Length().

9.52.3.13 UpnpStateVarComplete_get_StateVarName_cstr()

```
const char * UpnpStateVarComplete_get_StateVarName_cstr (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_StateVarName_cstr

References UpnpStateVarComplete_get_StateVarName(), and UpnpString_get_String().

9.52.3.14 UpnpStateVarComplete_get_StateVarName_Length()

```
size_t UpnpStateVarComplete_get_StateVarName_Length (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_StateVarName_Length

References UpnpStateVarComplete_get_StateVarName(), and UpnpString_get_Length().

9.52.3.15 UpnpStateVarComplete_new()

```
UpnpStateVarComplete * UpnpStateVarComplete_new ( )
```

Constructor

References UpnpString_new().

Referenced by UpnpStateVarComplete_dup(), and UpnpThreadDistribution().

9.52.3.16 UpnpStateVarComplete_set_CtrlUrl()

```
int UpnpStateVarComplete_set_CtrlUrl (
            UpnpStateVarComplete * p,
            const UpnpString * s )
```

UpnpStateVarComplete_set_CtrlUrl

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarComplete_assign().

9.52.3.17 UpnpStateVarComplete_set_CurrentVal()

```
int UpnpStateVarComplete_set_CurrentVal (
            UpnpStateVarComplete * p,
            const DOMString s )
```

UpnpStateVarComplete_set_CurrentVal

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by UpnpStateVarComplete_assign(), and UpnpThreadDistribution().

9.52.3.18 UpnpStateVarComplete_set_ErrCode()

```
int UpnpStateVarComplete_set_ErrCode (
            UpnpStateVarComplete * p,
            int n )
```

UpnpStateVarComplete_set_ErrCode

Referenced by UpnpStateVarComplete_assign(), and UpnpThreadDistribution().

9.52.3.19 UpnpStateVarComplete_set_StateVarName()

```
int UpnpStateVarComplete_set_StateVarName (
            UpnpStateVarComplete * p,
            const UpnpString * s )
```

UpnpStateVarComplete_set_StateVarName

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarComplete_assign().

9.52.3.20 UpnpStateVarComplete_strcpy_CtrlUrl()

```
int UpnpStateVarComplete_strcpy_CtrlUrl (
            UpnpStateVarComplete * p,
            const char * s )
```

UpnpStateVarComplete_strcpy_CtrlUrl

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.52.3.21 UpnpStateVarComplete_strcpy_StateVarName()

```
int UpnpStateVarComplete_strcpy_StateVarName (
            UpnpStateVarComplete * p,
            const char * s )
```

UpnpStateVarComplete_strcpy_StateVarName

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.52.3.22 UpnpStateVarComplete_strncpy_CtrlUrl()

```
int UpnpStateVarComplete_strncpy_CtrlUrl (
            UpnpStateVarComplete * p,
            const char * s,
            size_t n )
```

UpnpStateVarComplete_strncpy_CtrlUrl

References UpnpString_set_StringN().

9.52.3.23 UpnpStateVarComplete_strncpy_StateVarName()

```
int UpnpStateVarComplete_strncpy_StateVarName (
            UpnpStateVarComplete * p,
            const char * s,
            size_t n )
```

UpnpStateVarComplete_strncpy_StateVarName

References UpnpString_set_StringN().

9.53 UpnpStateVarComplete.h

Go to the documentation of this file.

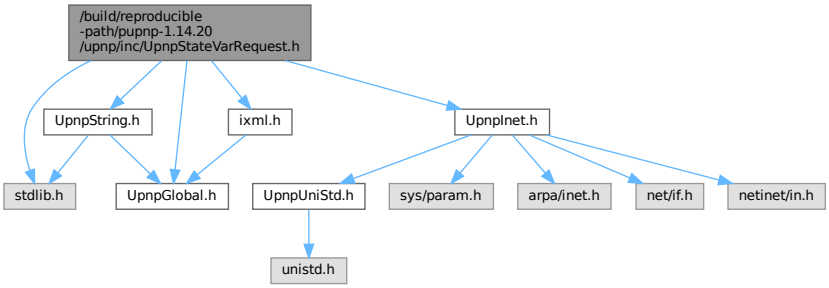
```
00001 #ifndef UPNPSTATEVARCOMPLETE_H
00002 #define UPNPSTATEVARCOMPLETE_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpString.h"
00019 #include "ixml.h"
00020
00021 #ifdef __cplusplus
00022 extern "C" {
00023 #endif /* __cplusplus */
00024
00028 typedef struct s_UpnpStateVarComplete UpnpStateVarComplete;
00029
00031 UPNP_EXPORT_SPEC UpnpStateVarComplete *UpnpStateVarComplete_new();
00033 UPNP_EXPORT_SPEC void UpnpStateVarComplete_delete(UpnpStateVarComplete *p);
00035 UPNP_EXPORT_SPEC UpnpStateVarComplete *UpnpStateVarComplete_dup(
00036         const UpnpStateVarComplete *p);
00038 UPNP_EXPORT_SPEC int UpnpStateVarComplete_assign(
00039         UpnpStateVarComplete *p, const UpnpStateVarComplete *q);
00040
00042 UPNP_EXPORT_SPEC int UpnpStateVarComplete_get_ErrCode(
00043         const UpnpStateVarComplete *p);
00045 UPNP_EXPORT_SPEC int UpnpStateVarComplete_set_ErrCode(
00046         UpnpStateVarComplete *p, int n);
00047
00049 UPNP_EXPORT_SPEC const UpnpString *UpnpStateVarComplete_get_CtrlUrl(
00050         const UpnpStateVarComplete *p);
00052 UPNP_EXPORT_SPEC int UpnpStateVarComplete_set_CtrlUrl(
00053         UpnpStateVarComplete *p, const UpnpString *s);
00055 UPNP_EXPORT_SPEC size_t UpnpStateVarComplete_get_CtrlUrl_Length(
00056         const UpnpStateVarComplete *p);
00058 UPNP_EXPORT_SPEC const char *UpnpStateVarComplete_get_CtrlUrl_cstr(
00059         const UpnpStateVarComplete *p);
00061 UPNP_EXPORT_SPEC int UpnpStateVarComplete_strcpy_CtrlUrl(
00062         UpnpStateVarComplete *p, const char *s);
00064 UPNP_EXPORT_SPEC int UpnpStateVarComplete_strncpy_CtrlUrl(
00065         UpnpStateVarComplete *p, const char *s, size_t n);
00067 UPNP_EXPORT_SPEC void UpnpStateVarComplete_clear_CtrlUrl(
00068         UpnpStateVarComplete *p);
00069
00071 UPNP_EXPORT_SPEC const UpnpString *UpnpStateVarComplete_get_StateVarName(
00072         const UpnpStateVarComplete *p);
00074 UPNP_EXPORT_SPEC int UpnpStateVarComplete_set_StateVarName(
00075         UpnpStateVarComplete *p, const UpnpString *s);
00077 UPNP_EXPORT_SPEC size_t UpnpStateVarComplete_get_StateVarName_Length(
00078         const UpnpStateVarComplete *p);
00080 UPNP_EXPORT_SPEC const char *UpnpStateVarComplete_get_StateVarName_cstr(
00081         const UpnpStateVarComplete *p);
00083 UPNP_EXPORT_SPEC int UpnpStateVarComplete_strcpy_StateVarName(
00084         UpnpStateVarComplete *p, const char *s);
00086 UPNP_EXPORT_SPEC int UpnpStateVarComplete_strncpy_StateVarName(
00087         UpnpStateVarComplete *p, const char *s, size_t n);
00089 UPNP_EXPORT_SPEC void UpnpStateVarComplete_clear_StateVarName(
00090         UpnpStateVarComplete *p);
00091
00093 UPNP_EXPORT_SPEC const DOMString UpnpStateVarComplete_get_CurrentVal(
00094         const UpnpStateVarComplete *p);
00096 UPNP_EXPORT_SPEC int UpnpStateVarComplete_set_CurrentVal(
00097         UpnpStateVarComplete *p, const DOMString s);
00099 UPNP_EXPORT_SPEC const char *UpnpStateVarComplete_get_CurrentVal_cstr(
00100         const UpnpStateVarComplete *p);
00101
00102 #ifdef __cplusplus
00103 }
00104 #endif /* __cplusplus */
00105
00106 #endif /* UPNPSTATEVARCOMPLETE_H */
```

9.54 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarRequest.h File Reference

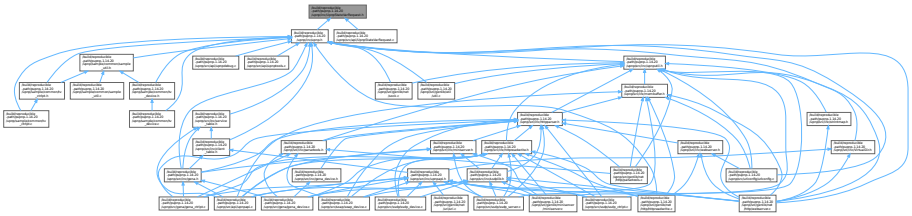
Header file for UpnpStateVarRequest methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include "UpnpString.h"
#include "ixml.h"
```

Include dependency graph for UpnpStateVarRequest.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpStateVarRequest UpnpStateVarRequest

Functions

- UpnpStateVarRequest * UpnpStateVarRequest_new ()
- void UpnpStateVarRequest_delete (UpnpStateVarRequest *p)
- UpnpStateVarRequest * UpnpStateVarRequest_dup (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_assign (UpnpStateVarRequest *p, const UpnpStateVarRequest *q)
- int UpnpStateVarRequest_get_ErrCode (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_ErrCode (UpnpStateVarRequest *p, int n)
- int UpnpStateVarRequest_get_Socket (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_Socket (UpnpStateVarRequest *p, int n)
- const UpnpString * UpnpStateVarRequest_get_ErrStr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_ErrStr (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_ErrStr_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_ErrStr_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_ErrStr (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_ErrStr (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_ErrStr (UpnpStateVarRequest *p)

- const UpnpString * UpnpStateVarRequest_get_DevUDN (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_DevUDN (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_DevUDN_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_DevUDN_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_DevUDN (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_DevUDN (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_DevUDN (UpnpStateVarRequest *p)
- const UpnpString * UpnpStateVarRequest_get_ServiceID (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_ServiceID (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_ServiceID_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_ServiceID_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_ServiceID (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_ServiceID (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_ServiceID (UpnpStateVarRequest *p)
- const UpnpString * UpnpStateVarRequest_get_StateVarName (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_StateVarName (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_StateVarName_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_StateVarName_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_StateVarName (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_StateVarName (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_StateVarName (UpnpStateVarRequest *p)
- const struct sockaddr_storage * UpnpStateVarRequest_get_CtrlPtIPAddr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_CtrlPtIPAddr (UpnpStateVarRequest *p, const struct sockaddr_storage *buf)
- void UpnpStateVarRequest_clear_CtrlPtIPAddr (UpnpStateVarRequest *p)
- const DOMString UpnpStateVarRequest_get_CurrentVal (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_CurrentVal (UpnpStateVarRequest *p, const DOMString s)
- const char * UpnpStateVarRequest_get_CurrentVal_cstr (const UpnpStateVarRequest *p)

9.54.1 Detailed Description

Header file for UpnpStateVarRequest methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.54.2 Typedef Documentation

9.54.2.1 UpnpStateVarRequest

```
typedef struct s_UpnpStateVarRequest UpnpStateVarRequest
```

UpnpStateVarRequest

9.54.3 Function Documentation

9.54.3.1 UpnpStateVarRequest_assign()

```
int UpnpStateVarRequest_assign (
            UpnpStateVarRequest * p,
            const UpnpStateVarRequest * q )
```

Assignment operator

References UpnpStateVarRequest_get_CtrlPtIPAddr(), UpnpStateVarRequest_get_CurrentVal(), UpnpStateVarRequest_get_DevUDI
UpnpStateVarRequest_get_ErrCode(), UpnpStateVarRequest_get_ErrStr(), UpnpStateVarRequest_get_ServiceID(),
UpnpStateVarRequest_get_Socket(), UpnpStateVarRequest_get_StateVarName(), UpnpStateVarRequest_set_CtrlPtIPAddr(),
UpnpStateVarRequest_set_CurrentVal(), UpnpStateVarRequest_set_DevUDN(), UpnpStateVarRequest_set_ErrCode(),
UpnpStateVarRequest_set_ErrStr(), UpnpStateVarRequest_set_ServiceID(), UpnpStateVarRequest_set_Socket(),
and UpnpStateVarRequest_set_StateVarName().

Referenced by UpnpStateVarRequest_dup().

9.54.3.2 UpnpStateVarRequest_clear_CtrlPtIPAddr()

```
void UpnpStateVarRequest_clear_CtrlPtIPAddr (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CtrlPtIPAddr

9.54.3.3 UpnpStateVarRequest_clear_DevUDN()

```
void UpnpStateVarRequest_clear_DevUDN (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_DevUDN

References UpnpString_clear().

9.54.3.4 UpnpStateVarRequest_clear_ErrStr()

```
void UpnpStateVarRequest_clear_ErrStr (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_ErrStr

References UpnpString_clear().

9.54.3.5 UpnpStateVarRequest_clear_ServiceID()

```
void UpnpStateVarRequest_clear_ServiceID (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_ServiceID

References UpnpString_clear().

9.54.3.6 UpnpStateVarRequest_clear_StateVarName()

```
void UpnpStateVarRequest_clear_StateVarName (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_StateVarName

References UpnpString_clear().

9.54.3.7 UpnpStateVarRequest_delete()

```
void UpnpStateVarRequest_delete (
            UpnpStateVarRequest * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by handle_query_variable().

9.54.3.8 UpnpStateVarRequest_dup()

```
UpnpStateVarRequest * UpnpStateVarRequest_dup (
            const UpnpStateVarRequest * p )
```

Copy Constructor

References UpnpStateVarRequest_assign(), and UpnpStateVarRequest_new().

9.54.3.9 UpnpStateVarRequest_get_CtrlPtIPAddr()

```
const struct sockaddr_storage * UpnpStateVarRequest_get_CtrlPtIPAddr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CtrlPtIPAddr

Referenced by UpnpStateVarRequest_assign().

9.54.3.10 UpnpStateVarRequest_get_CurrentVal()

```
const DOMString UpnpStateVarRequest_get_CurrentVal (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CurrentVal

Referenced by handle_query_variable(), SampleUtil_PrintEvent(), UpnpStateVarRequest_assign(), and UpnpStateVarRequest_get_CurrentVal_cstr().

9.54.3.11 UpnpStateVarRequest_get_CurrentVal_cstr()

```
const char * UpnpStateVarRequest_get_CurrentVal_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CurrentVal_cstr

References UpnpStateVarRequest_get_CurrentVal().

9.54.3.12 UpnpStateVarRequest_get_DevUDN()

```
const UpnpString * UpnpStateVarRequest_get_DevUDN (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_DevUDN

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_DevUDN_cstr(), and UpnpStateVarRequest_get_DevUDN_Length().

9.54.3.13 UpnpStateVarRequest_get_DevUDN_cstr()

```
const char * UpnpStateVarRequest_get_DevUDN_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_DevUDN_cstr

References UpnpStateVarRequest_get_DevUDN(), and UpnpString_get_String().

9.54.3.14 UpnpStateVarRequest_get_DevUDN_Length()

```
size_t UpnpStateVarRequest_get_DevUDN_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_DevUDN_Length

References UpnpStateVarRequest_get_DevUDN(), and UpnpString_get_Length().

9.54.3.15 UpnpStateVarRequest_get_ErrCode()

```
int UpnpStateVarRequest_get_ErrCode (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrCode

Referenced by handle_query_variable(), SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), and UpnpStateVarRequest_assign().

9.54.3.16 UpnpStateVarRequest_get_ErrStr()

```
const UpnpString * UpnpStateVarRequest_get_ErrStr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrStr

Referenced by handle_query_variable(), SampleUtil_PrintEvent(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_ErrStr and UpnpStateVarRequest_get_ErrStr_Length().

9.54.3.17 UpnpStateVarRequest_get_ErrStr_cstr()

```
const char * UpnpStateVarRequest_get_ErrStr_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrStr_cstr

References UpnpStateVarRequest_get_ErrStr(), and UpnpString_get_String().

Referenced by handle_query_variable().

9.54.3.18 UpnpStateVarRequest_get_ErrStr_Length()

```
size_t UpnpStateVarRequest_get_ErrStr_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrStr_Length

References UpnpStateVarRequest_get_ErrStr(), and UpnpString_get_Length().

9.54.3.19 UpnpStateVarRequest_get_ServiceID()

```
const UpnpString * UpnpStateVarRequest_get_ServiceID (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ServiceID

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_ServiceID_cstr(), and UpnpStateVarRequest_get_ServiceID_Length().

9.54.3.20 UpnpStateVarRequest_get_ServiceID_cstr()

```
const char * UpnpStateVarRequest_get_ServiceID_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ServiceID_cstr

References UpnpStateVarRequest_get_ServiceID(), and UpnpString_get_String().

9.54.3.21 UpnpStateVarRequest_get_ServiceID_Length()

```
size_t UpnpStateVarRequest_get_ServiceID_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ServiceID_Length

References UpnpStateVarRequest_get_ServiceID(), and UpnpString_get_Length().

9.54.3.22 UpnpStateVarRequest_get_Socket()

```
int UpnpStateVarRequest_get_Socket (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_Socket

Referenced by UpnpStateVarRequest_assign().

9.54.3.23 UpnpStateVarRequest_get_StateVarName()

```
const UpnpString * UpnpStateVarRequest_get_StateVarName (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_StateVarName

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_StateVarName_cstr(), and UpnpStateVarRequest_get_StateVarName_Length().

9.54.3.24 UpnpStateVarRequest_get_StateVarName_cstr()

```
const char * UpnpStateVarRequest_get_StateVarName_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_StateVarName_cstr

References UpnpStateVarRequest_get_StateVarName(), and UpnpString_get_String().

9.54.3.25 UpnpStateVarRequest_get_StateVarName_Length()

```
size_t UpnpStateVarRequest_get_StateVarName_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_StateVarName_Length

References UpnpStateVarRequest_get_StateVarName(), and UpnpString_get_Length().

9.54.3.26 UpnpStateVarRequest_new()

```
UpnpStateVarRequest * UpnpStateVarRequest_new ( )
```

Constructor

References UpnpString_new().

Referenced by handle_query_variable(), and UpnpStateVarRequest_dup().

9.54.3.27 UpnpStateVarRequest_set_CtrlPtIPAddr()

```
int UpnpStateVarRequest_set_CtrlPtIPAddr (
            UpnpStateVarRequest * p,
            const struct sockaddr_storage * buf )
```

UpnpStateVarRequest_get_CtrlPtIPAddr

Referenced by handle_query_variable(), and UpnpStateVarRequest_assign().

9.54.3.28 UpnpStateVarRequest_set_CurrentVal()

```
int UpnpStateVarRequest_set_CurrentVal (
            UpnpStateVarRequest * p,
            const DOMString s )
```

UpnpStateVarRequest_set_CurrentVal

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by TvDeviceHandleGetVarRequest(), and UpnpStateVarRequest_assign().

9.54.3.29 UpnpStateVarRequest_set_DevUDN()

```
int UpnpStateVarRequest_set_DevUDN (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_DevUDN

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.54.3.30 UpnpStateVarRequest_set_ErrCode()

```
int UpnpStateVarRequest_set_ErrCode (
            UpnpStateVarRequest * p,
            int n )
```

UpnpStateVarRequest_set_ErrCode

Referenced by handle_query_variable(), TvDeviceHandleGetVarRequest(), and UpnpStateVarRequest_assign().

9.54.3.31 UpnpStateVarRequest_set_ErrStr()

```
int UpnpStateVarRequest_set_ErrStr (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_ErrStr

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.54.3.32 UpnpStateVarRequest_set_ServiceID()

```
int UpnpStateVarRequest_set_ServiceID (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_ServiceID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.54.3.33 UpnpStateVarRequest_set_Socket()

```
int UpnpStateVarRequest_set_Socket (
            UpnpStateVarRequest * p,
            int n )
```

UpnpStateVarRequest_set_Socket

Referenced by UpnpStateVarRequest_assign().

9.54.3.34 UpnpStateVarRequest_set_StateVarName()

```
int UpnpStateVarRequest_set_StateVarName (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_StateVarName

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.54.3.35 UpnpStateVarRequest_strcpy_DevUDN()

```
int UpnpStateVarRequest_strcpy_DevUDN (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_DevUDN

References UpnpString_set_String().

Referenced by handle_query_variable().

9.54.3.36 UpnpStateVarRequest_strcpy_ErrStr()

```
int UpnpStateVarRequest_strcpy_ErrStr (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_ErrStr

References UpnpString_set_String().

Referenced by TvDeviceHandleGetVarRequest().

9.54.3.37 UpnpStateVarRequest_strcpy_ServiceID()

```
int UpnpStateVarRequest_strcpy_ServiceID (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_ServiceID

References UpnpString_set_String().

Referenced by handle_query_variable().

9.54.3.38 UpnpStateVarRequest_strcpy_StateVarName()

```
int UpnpStateVarRequest_strcpy_StateVarName (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_StateVarName

References UpnpString_set_String().

Referenced by handle_query_variable().

9.54.3.39 UpnpStateVarRequest_strncpy_DevUDN()

```
int UpnpStateVarRequest_strncpy_DevUDN (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_DevUDN

References UpnpString_set_StringN().

9.54.3.40 UpnpStateVarRequest_strncpy_ErrStr()

```
int UpnpStateVarRequest_strncpy_ErrStr (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_ErrStr

References UpnpString_set_StringN().

9.54.3.41 UpnpStateVarRequest_strncpy_ServiceID()

```
int UpnpStateVarRequest_strncpy_ServiceID (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_ServiceID

References UpnpString_set_StringN().

9.54.3.42 UpnpStateVarRequest_strncpy_StateVarName()

```
int UpnpStateVarRequest_strncpy_StateVarName (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_StateVarName

References UpnpString_set_StringN().

9.55 UpnpStateVarRequest.h

Go to the documentation of this file.

```
00001 #ifndef UPNPSTATEVARREQUEST_H
00002 #define UPNPSTATEVARREQUEST_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpInet.h"
00019 #include "UpnpString.h"
00020 #include "ixml.h"
00021
00022 #ifdef __cplusplus
00023 extern "C" {
00024 #endif /* __cplusplus */
00025
00029 typedef struct s_UpnpStateVarRequest UpnpStateVarRequest;
00030
00032 UPNP_EXPORT_SPEC UpnpStateVarRequest *UpnpStateVarRequest_new();
00034 UPNP_EXPORT_SPEC void UpnpStateVarRequest_delete(UpnpStateVarRequest *p);
00036 UPNP_EXPORT_SPEC UpnpStateVarRequest *UpnpStateVarRequest_dup(
00037         const UpnpStateVarRequest *p);
00039 UPNP_EXPORT_SPEC int UpnpStateVarRequest_assign(
00040         UpnpStateVarRequest *p, const UpnpStateVarRequest *q);
00041
00043 UPNP_EXPORT_SPEC int UpnpStateVarRequest_get_ErrCode(
00044         const UpnpStateVarRequest *p);
00046 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_ErrCode(
00047         UpnpStateVarRequest *p, int n);
00048
00050 UPNP_EXPORT_SPEC int UpnpStateVarRequest_get_Socket(
00051         const UpnpStateVarRequest *p);
00053 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_Socket(
00054         UpnpStateVarRequest *p, int n);
00055
00057 UPNP_EXPORT_SPEC const UpnpString *UpnpStateVarRequest_get_ErrStr(
00058         const UpnpStateVarRequest *p);
00060 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_ErrStr(
00061         UpnpStateVarRequest *p, const UpnpString *s);
00063 UPNP_EXPORT_SPEC size_t UpnpStateVarRequest_get_ErrStr_Length(
00064         const UpnpStateVarRequest *p);
00066 UPNP_EXPORT_SPEC const char *UpnpStateVarRequest_get_ErrStr_cstr(
00067         const UpnpStateVarRequest *p);
00069 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strcpy_ErrStr(
00070         UpnpStateVarRequest *p, const char *s);
00072 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strncpy_ErrStr(
00073         UpnpStateVarRequest *p, const char *s, size_t n);
00075 UPNP_EXPORT_SPEC void UpnpStateVarRequest_clear_ErrStr(UpnpStateVarRequest *p);
00076
00078 UPNP_EXPORT_SPEC const UpnpString *UpnpStateVarRequest_get_DevUDN(
00079         const UpnpStateVarRequest *p);
00081 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_DevUDN(
00082         UpnpStateVarRequest *p, const UpnpString *s);
00084 UPNP_EXPORT_SPEC size_t UpnpStateVarRequest_get_DevUDN_Length(
00085         const UpnpStateVarRequest *p);
00087 UPNP_EXPORT_SPEC const char *UpnpStateVarRequest_get_DevUDN_cstr(
00088         const UpnpStateVarRequest *p);
00090 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strcpy_DevUDN(
00091         UpnpStateVarRequest *p, const char *s);
00093 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strncpy_DevUDN(
00094         UpnpStateVarRequest *p, const char *s, size_t n);
00096 UPNP_EXPORT_SPEC void UpnpStateVarRequest_clear_DevUDN(UpnpStateVarRequest *p);
00097
00099 UPNP_EXPORT_SPEC const UpnpString *UpnpStateVarRequest_get_ServiceID(
00100         const UpnpStateVarRequest *p);
00102 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_ServiceID(
00103         UpnpStateVarRequest *p, const UpnpString *s);
00105 UPNP_EXPORT_SPEC size_t UpnpStateVarRequest_get_ServiceID_Length(
00106         const UpnpStateVarRequest *p);
00108 UPNP_EXPORT_SPEC const char *UpnpStateVarRequest_get_ServiceID_cstr(
00109         const UpnpStateVarRequest *p);
00111 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strcpy_ServiceID(
00112         UpnpStateVarRequest *p, const char *s);
00114 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strncpy_ServiceID(
00115         UpnpStateVarRequest *p, const char *s, size_t n);
00117 UPNP_EXPORT_SPEC void UpnpStateVarRequest_clear_ServiceID(
00118         UpnpStateVarRequest *p);
00119
00121 UPNP_EXPORT_SPEC const UpnpString *UpnpStateVarRequest_get_StateVarName(
00122         const UpnpStateVarRequest *p);
00124 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_StateVarName(
00125         UpnpStateVarRequest *p, const UpnpString *s);
00127 UPNP_EXPORT_SPEC size_t UpnpStateVarRequest_get_StateVarName_Length(
```

```
00128         const UpnpStateVarRequest *p);
00130 UPNP_EXPORT_SPEC const char *UpnpStateVarRequest_get_StateVarName_cstr(
00131         const UpnpStateVarRequest *p);
00133 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strcpy_StateVarName(
00134         UpnpStateVarRequest *p, const char *s);
00136 UPNP_EXPORT_SPEC int UpnpStateVarRequest_strncpy_StateVarName(
00137         UpnpStateVarRequest *p, const char *s, size_t n);
00139 UPNP_EXPORT_SPEC void UpnpStateVarRequest_clear_StateVarName(
00140         UpnpStateVarRequest *p);
00141
00143 UPNP_EXPORT_SPEC const struct sockaddr_storage *
00144 UpnpStateVarRequest_get_CtrlPtIPAddr(const UpnpStateVarRequest *p);
00146 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_CtrlPtIPAddr(
00147         UpnpStateVarRequest *p, const struct sockaddr_storage *buf);
00149 UPNP_EXPORT_SPEC void UpnpStateVarRequest_clear_CtrlPtIPAddr(
00150         UpnpStateVarRequest *p);
00151
00153 UPNP_EXPORT_SPEC const DOMString UpnpStateVarRequest_get_CurrentVal(
00154         const UpnpStateVarRequest *p);
00156 UPNP_EXPORT_SPEC int UpnpStateVarRequest_set_CurrentVal(
00157         UpnpStateVarRequest *p, const DOMString s);
00159 UPNP_EXPORT_SPEC const char *UpnpStateVarRequest_get_CurrentVal_cstr(
00160         const UpnpStateVarRequest *p);
00161
00162 #ifdef __cplusplus
00163 }
00164 #endif /* __cplusplus */
00165
00166 #endif /* UPNPSTATEVARREQUEST_H */
```

9.56 UpnpStdInt.h

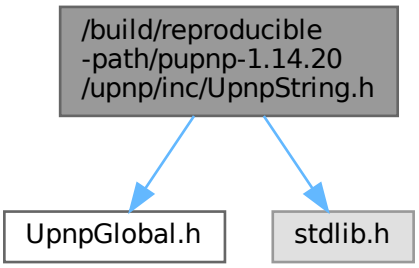
```
00001 #ifndef UPNPSTDINT_H
00002 #define UPNPSTDINT_H
00003
00004 /* Sized integer types. */
00005 #include <stdint.h>
00006
00007 #if !defined(UPNP_USE_BCBPP)
00008
00009         #ifdef UPNP_USE_MSVCPP
00010                 /* no ssize_t defined for VC */
00011                 #ifdef _WIN64
00012 typedef int64_t ssize_t;
00013                 #else
00014 typedef int32_t ssize_t;
00015                 #endif
00016         #endif
00017
00018 #endif /* !defined(UPNP_USE_BCBPP) */
00019
00020 #endif /* UPNPSTDINT_H */
```

9.57 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpString.h File Reference

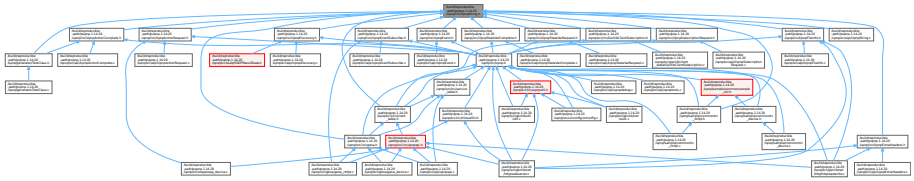
UpnpString object declaration.

```
#include "UpnpGlobal.h"
#include <stdlib.h>
```

Include dependency graph for UpnpString.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpString UpnpString

  Type of the string objects inside libupnp.

Functions

- UpnpString * UpnpString_new (void)

  Constructor.
- void UpnpString_delete (UpnpString *p)

  Destructor.
- UpnpString * UpnpString_dup (const UpnpString *p)

  Copy Constructor.
- void UpnpString_assign (UpnpString *p, const UpnpString *q)

  Assignment operator.
- size_t UpnpString_get_Length (const UpnpString *p)

  Returns the length of the string.
- void UpnpString_set_Length (UpnpString *p, size_t n)

  Truncates the string to the specified lenght, or does nothing if the current lenght is less than or equal to the requested length.
- const char * UpnpString_get_String (const UpnpString *p)

  Returns the pointer to char.
- int UpnpString_set_String (UpnpString *p, const char *s)

  Sets the string from a pointer to char.

- int UpnpString_set_StringN (UpnpString *p, const char *s, size_t n)

  Sets the string from a pointer to char using a maximum of N chars.
- void UpnpString_clear (UpnpString *p)

  Clears the string, sets its size to zero.
- int UpnpString_cmp (UpnpString *p, UpnpString *q)

  Compares two strings for equality. Case matters.
- int UpnpString_casecmp (UpnpString *p, UpnpString *q)

  Compares two strings for equality. Case does not matter.

9.57.1 Detailed Description

UpnpString object declaration.

9.58 UpnpString.h

Go to the documentation of this file.

```
00001
00002
00003 #ifndef STRING_H
00004 #define STRING_H
00005
00022 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00023
00024 #include <stdlib.h> /* for size_t */
00025
00026 #ifdef __cplusplus
00027 extern "C" {
00028 #endif /* __cplusplus */
00029
00033 typedef struct s_UpnpString UpnpString;
00034
00040 UPNP_EXPORT_SPEC UpnpString *UpnpString_new(void);
00041
00045 UPNP_EXPORT_SPEC void UpnpString_delete(
00047         UpnpString *p);
00048
00054 UPNP_EXPORT_SPEC UpnpString *UpnpString_dup(
00056         const UpnpString *p);
00057
00061 UPNP_EXPORT_SPEC void UpnpString_assign(
00063         UpnpString *p,
00065         const UpnpString *q);
00066
00072 UPNP_EXPORT_SPEC size_t UpnpString_get_Length(
00074         const UpnpString *p);
00075
00080 UPNP_EXPORT_SPEC void UpnpString_set_Length(
00082         UpnpString *p,
00084         size_t n);
00085
00091 UPNP_EXPORT_SPEC const char *UpnpString_get_String(
00093         const UpnpString *p);
00094
00098 UPNP_EXPORT_SPEC int UpnpString_set_String(
00100         UpnpString *p,
00102         const char *s);
00103
00107 UPNP_EXPORT_SPEC int UpnpString_set_StringN(
00109         UpnpString *p,
00111         const char *s,
00113         size_t n);
00114
00118 UPNP_EXPORT_SPEC void UpnpString_clear(
00120         UpnpString *p);
00121
00127 UPNP_EXPORT_SPEC int UpnpString_cmp(
00129         UpnpString *p,
00131         UpnpString *q);
00132
00138 UPNP_EXPORT_SPEC int UpnpString_casecmp(
00140         UpnpString *p,
```

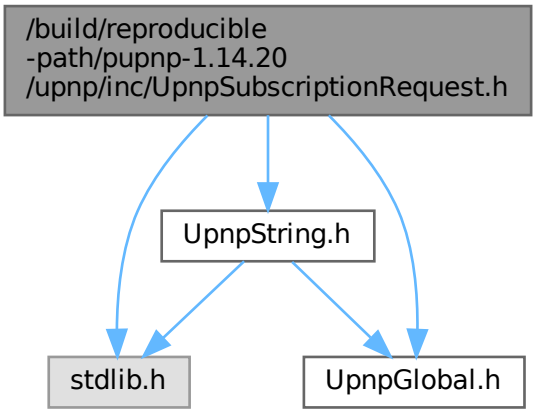
```
00142         UpnpString *q);
00143
00144 #ifdef __cplusplus
00145 }
00146 #endif /* __cplusplus */
00147
00148 /* @} UpnpString The UpnpString API */
00149
00150 #endif /* STRING_H */
```

9.59 /build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpSubscriptionRequest.h File Reference

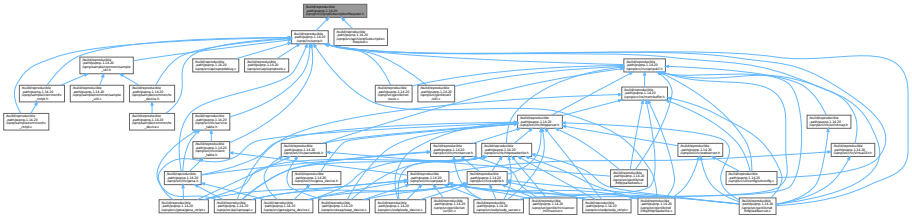
Header file for UpnpSubscriptionRequest methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpString.h"
```

Include dependency graph for UpnpSubscriptionRequest.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_UpnpSubscriptionRequest UpnpSubscriptionRequest

Functions

- UpnpSubscriptionRequest * UpnpSubscriptionRequest_new ()
- void UpnpSubscriptionRequest_delete (UpnpSubscriptionRequest *p)
- UpnpSubscriptionRequest * UpnpSubscriptionRequest_dup (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_assign (UpnpSubscriptionRequest *p, const UpnpSubscriptionRequest *q)
- const UpnpString * UpnpSubscriptionRequest_get_ServiceId (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_set_ServiceId (UpnpSubscriptionRequest *p, const UpnpString *s)
- size_t UpnpSubscriptionRequest_get_ServiceId_Length (const UpnpSubscriptionRequest *p)
- const char * UpnpSubscriptionRequest_get_ServiceId_cstr (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_strcpy_ServiceId (UpnpSubscriptionRequest *p, const char *s)
- int UpnpSubscriptionRequest_strncpy_ServiceId (UpnpSubscriptionRequest *p, const char *s, size_t n)
- void UpnpSubscriptionRequest_clear_ServiceId (UpnpSubscriptionRequest *p)
- const UpnpString * UpnpSubscriptionRequest_get_UDN (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_set_UDN (UpnpSubscriptionRequest *p, const UpnpString *s)
- size_t UpnpSubscriptionRequest_get_UDN_Length (const UpnpSubscriptionRequest *p)
- const char * UpnpSubscriptionRequest_get_UDN_cstr (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_strcpy_UDN (UpnpSubscriptionRequest *p, const char *s)
- int UpnpSubscriptionRequest_strncpy_UDN (UpnpSubscriptionRequest *p, const char *s, size_t n)
- void UpnpSubscriptionRequest_clear_UDN (UpnpSubscriptionRequest *p)
- const UpnpString * UpnpSubscriptionRequest_get_SID (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_set_SID (UpnpSubscriptionRequest *p, const UpnpString *s)
- size_t UpnpSubscriptionRequest_get_SID_Length (const UpnpSubscriptionRequest *p)
- const char * UpnpSubscriptionRequest_get_SID_cstr (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_strcpy_SID (UpnpSubscriptionRequest *p, const char *s)
- int UpnpSubscriptionRequest_strncpy_SID (UpnpSubscriptionRequest *p, const char *s, size_t n)
- void UpnpSubscriptionRequest_clear_SID (UpnpSubscriptionRequest *p)

9.59.1 Detailed Description

Header file for UpnpSubscriptionRequest methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.59.2 Typedef Documentation

9.59.2.1 UpnpSubscriptionRequest

```
typedef struct s_UpnpSubscriptionRequest UpnpSubscriptionRequest
```

UpnpSubscriptionRequest

9.59.3 Function Documentation

9.59.3.1 UpnpSubscriptionRequest_assign()

```
int UpnpSubscriptionRequest_assign (
            UpnpSubscriptionRequest * p,
            const UpnpSubscriptionRequest * q )
```

Assignment operator

References UpnpSubscriptionRequest_get_ServiceId(), UpnpSubscriptionRequest_get_SID(), UpnpSubscriptionRequest_get_UDN(), UpnpSubscriptionRequest_set_ServiceId(), UpnpSubscriptionRequest_set_SID(), and UpnpSubscriptionRequest_set_UDN().

Referenced by UpnpSubscriptionRequest_dup().

9.59.3.2 UpnpSubscriptionRequest_clear_ServiceId()

```
void UpnpSubscriptionRequest_clear_ServiceId (
            UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_clear_ServiceId

References UpnpString_clear().

9.59.3.3 UpnpSubscriptionRequest_clear_SID()

```
void UpnpSubscriptionRequest_clear_SID (
            UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_clear_SID

References UpnpString_clear().

9.59.3.4 UpnpSubscriptionRequest_clear_UDN()

```
void UpnpSubscriptionRequest_clear_UDN (
            UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_clear_UDN

References UpnpString_clear().

9.59.3.5 UpnpSubscriptionRequest_delete()

```
void UpnpSubscriptionRequest_delete (
            UpnpSubscriptionRequest * p )
```

Destructor

References UpnpString_delete().

Referenced by gena_process_subscription_request().

9.59.3.6 UpnpSubscriptionRequest_dup()

```
UpnpSubscriptionRequest * UpnpSubscriptionRequest_dup (
            const UpnpSubscriptionRequest * p )
```

Copy Constructor

References UpnpSubscriptionRequest_assign(), and UpnpSubscriptionRequest_new().

9.59.3.7 UpnpSubscriptionRequest_get_ServiceId()

```
const UpnpString * UpnpSubscriptionRequest_get_ServiceId (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_ServiceId

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleSubscriptionRequest(), UpnpSubscriptionRequest_assign(), UpnpSubscriptionRequest_get_ServiceId_cstr(), and UpnpSubscriptionRequest_get_ServiceId_Length().

9.59.3.8 UpnpSubscriptionRequest_get_ServiceId_cstr()

```
const char * UpnpSubscriptionRequest_get_ServiceId_cstr (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_ServiceId_cstr

References UpnpString_get_String(), and UpnpSubscriptionRequest_get_ServiceId().

9.59.3.9 UpnpSubscriptionRequest_get_ServiceId_Length()

```
size_t UpnpSubscriptionRequest_get_ServiceId_Length (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_ServiceId_Length

References UpnpString_get_Length(), and UpnpSubscriptionRequest_get_ServiceId().

9.59.3.10 UpnpSubscriptionRequest_get_SID()

```
const UpnpString * UpnpSubscriptionRequest_get_SID (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_SID

Referenced by SampleUtil_PrintEvent(), UpnpSubscriptionRequest_assign(), UpnpSubscriptionRequest_get_SID_cstr(), and UpnpSubscriptionRequest_get_SID_Length().

9.59.3.11 UpnpSubscriptionRequest_get_SID_cstr()

```
const char * UpnpSubscriptionRequest_get_SID_cstr (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_SID_cstr

References UpnpString_get_String(), and UpnpSubscriptionRequest_get_SID().

Referenced by TvDeviceHandleSubscriptionRequest().

9.59.3.12 UpnpSubscriptionRequest_get_SID_Length()

```
size_t UpnpSubscriptionRequest_get_SID_Length (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_SID_Length

References UpnpString_get_Length(), and UpnpSubscriptionRequest_get_SID().

9.59.3.13 UpnpSubscriptionRequest_get_UDN()

```
const UpnpString * UpnpSubscriptionRequest_get_UDN (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_UDN

Referenced by SampleUtil_PrintEvent(), UpnpSubscriptionRequest_assign(), UpnpSubscriptionRequest_get_UDN_cstr(), and UpnpSubscriptionRequest_get_UDN_Length().

9.59.3.14 UpnpSubscriptionRequest_get_UDN_cstr()

```
const char * UpnpSubscriptionRequest_get_UDN_cstr (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_UDN_cstr

References UpnpString_get_String(), and UpnpSubscriptionRequest_get_UDN().

Referenced by TvDeviceHandleSubscriptionRequest().

9.59.3.15 UpnpSubscriptionRequest_get_UDN_Length()

```
size_t UpnpSubscriptionRequest_get_UDN_Length (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_UDN_Length

References UpnpString_get_Length(), and UpnpSubscriptionRequest_get_UDN().

9.59.3.16 UpnpSubscriptionRequest_new()

```
UpnpSubscriptionRequest * UpnpSubscriptionRequest_new ( )
```

Constructor

References UpnpString_new().

Referenced by gena_process_subscription_request(), and UpnpSubscriptionRequest_dup().

9.59.3.17 UpnpSubscriptionRequest_set_ServiceId()

```
int UpnpSubscriptionRequest_set_ServiceId (
            UpnpSubscriptionRequest * p,
            const UpnpString * s )
```

UpnpSubscriptionRequest_set_ServiceId

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpSubscriptionRequest_assign().

9.59.3.18 UpnpSubscriptionRequest_set_SID()

```
int UpnpSubscriptionRequest_set_SID (
            UpnpSubscriptionRequest * p,
            const UpnpString * s )
```

UpnpSubscriptionRequest_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpSubscriptionRequest_assign().

9.59.3.19 UpnpSubscriptionRequest_set_UDN()

```
int UpnpSubscriptionRequest_set_UDN (
            UpnpSubscriptionRequest * p,
            const UpnpString * s )
```

UpnpSubscriptionRequest_set_UDN

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpSubscriptionRequest_assign().

9.59.3.20 UpnpSubscriptionRequest_strcpy_ServiceId()

```
int UpnpSubscriptionRequest_strcpy_ServiceId (
            UpnpSubscriptionRequest * p,
            const char * s )
```

UpnpSubscriptionRequest_strcpy_ServiceId

References UpnpString_set_String().

Referenced by gena_process_subscription_request().

9.59.3.21 UpnpSubscriptionRequest_strcpy_SID()

```
int UpnpSubscriptionRequest_strcpy_SID (
            UpnpSubscriptionRequest * p,
            const char * s )
```

UpnpSubscriptionRequest_strcpy_SID

References UpnpString_set_String().

Referenced by gena_process_subscription_request().

9.59.3.22 UpnpSubscriptionRequest_strcpy_UDN()

```
int UpnpSubscriptionRequest_strcpy_UDN (
            UpnpSubscriptionRequest * p,
            const char * s )
```

UpnpSubscriptionRequest_strcpy_UDN

References UpnpString_set_String().

Referenced by gena_process_subscription_request().

9.59.3.23 UpnpSubscriptionRequest_strncpy_ServiceId()

```
int UpnpSubscriptionRequest_strncpy_ServiceId (
            UpnpSubscriptionRequest * p,
            const char * s,
            size_t n )
```

UpnpSubscriptionRequest_strncpy_ServiceId

References UpnpString_set_StringN().

9.59.3.24 UpnpSubscriptionRequest_strncpy_SID()

```
int UpnpSubscriptionRequest_strncpy_SID (
            UpnpSubscriptionRequest * p,
            const char * s,
            size_t n )
```

UpnpSubscriptionRequest_strncpy_SID

References UpnpString_set_StringN().

9.59.3.25 UpnpSubscriptionRequest_strncpy_UDN()

```
int UpnpSubscriptionRequest_strncpy_UDN (
            UpnpSubscriptionRequest * p,
            const char * s,
            size_t n )
```

UpnpSubscriptionRequest_strncpy_UDN

References UpnpString_set_StringN().

9.60 UpnpSubscriptionRequest.h

Go to the documentation of this file.

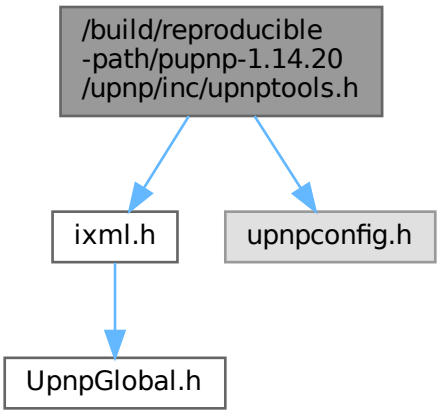
```
00001 #ifndef UPNPSUBSCRIPTIONREQUEST_H
00002 #define UPNPSUBSCRIPTIONREQUEST_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpString.h"
00019
00020 #ifdef __cplusplus
00021 extern "C" {
00022 #endif /* __cplusplus */
00023
00027 typedef struct s_UpnpSubscriptionRequest UpnpSubscriptionRequest;
00028
00030 UPNP_EXPORT_SPEC UpnpSubscriptionRequest *UpnpSubscriptionRequest_new();
00032 UPNP_EXPORT_SPEC void UpnpSubscriptionRequest_delete(
00033         UpnpSubscriptionRequest *p);
00035 UPNP_EXPORT_SPEC UpnpSubscriptionRequest *UpnpSubscriptionRequest_dup(
00036         const UpnpSubscriptionRequest *p);
00038 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_assign(
00039         UpnpSubscriptionRequest *p, const UpnpSubscriptionRequest *q);
00040
00042 UPNP_EXPORT_SPEC const UpnpString *UpnpSubscriptionRequest_get_ServiceId(
00043         const UpnpSubscriptionRequest *p);
00045 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_set_ServiceId(
00046         UpnpSubscriptionRequest *p, const UpnpString *s);
00048 UPNP_EXPORT_SPEC size_t UpnpSubscriptionRequest_get_ServiceId_Length(
00049         const UpnpSubscriptionRequest *p);
00051 UPNP_EXPORT_SPEC const char *UpnpSubscriptionRequest_get_ServiceId_cstr(
00052         const UpnpSubscriptionRequest *p);
00054 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_strcpy_ServiceId(
00055         UpnpSubscriptionRequest *p, const char *s);
00057 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_strncpy_ServiceId(
00058         UpnpSubscriptionRequest *p, const char *s, size_t n);
00060 UPNP_EXPORT_SPEC void UpnpSubscriptionRequest_clear_ServiceId(
00061         UpnpSubscriptionRequest *p);
00062
00064 UPNP_EXPORT_SPEC const UpnpString *UpnpSubscriptionRequest_get_UDN(
00065         const UpnpSubscriptionRequest *p);
00067 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_set_UDN(
00068         UpnpSubscriptionRequest *p, const UpnpString *s);
00070 UPNP_EXPORT_SPEC size_t UpnpSubscriptionRequest_get_UDN_Length(
```

```
00071         const UpnpSubscriptionRequest *p);
00073 UPNP_EXPORT_SPEC const char *UpnpSubscriptionRequest_get_UDN_cstr(
00074         const UpnpSubscriptionRequest *p);
00076 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_strcpy_UDN(
00077         UpnpSubscriptionRequest *p, const char *s);
00079 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_strncpy_UDN(
00080         UpnpSubscriptionRequest *p, const char *s, size_t n);
00082 UPNP_EXPORT_SPEC void UpnpSubscriptionRequest_clear_UDN(
00083         UpnpSubscriptionRequest *p);
00084
00086 UPNP_EXPORT_SPEC const UpnpString *UpnpSubscriptionRequest_get_SID(
00087         const UpnpSubscriptionRequest *p);
00089 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_set_SID(
00090         UpnpSubscriptionRequest *p, const UpnpString *s);
00092 UPNP_EXPORT_SPEC size_t UpnpSubscriptionRequest_get_SID_Length(
00093         const UpnpSubscriptionRequest *p);
00095 UPNP_EXPORT_SPEC const char *UpnpSubscriptionRequest_get_SID_cstr(
00096         const UpnpSubscriptionRequest *p);
00098 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_strcpy_SID(
00099         UpnpSubscriptionRequest *p, const char *s);
00101 UPNP_EXPORT_SPEC int UpnpSubscriptionRequest_strncpy_SID(
00102         UpnpSubscriptionRequest *p, const char *s, size_t n);
00104 UPNP_EXPORT_SPEC void UpnpSubscriptionRequest_clear_SID(
00105         UpnpSubscriptionRequest *p);
00106
00107 #ifdef __cplusplus
00108 }
00109 #endif /* __cplusplus */
00110
00111 #endif /* UPNPSUBSCRIPTIONREQUEST_H */
```

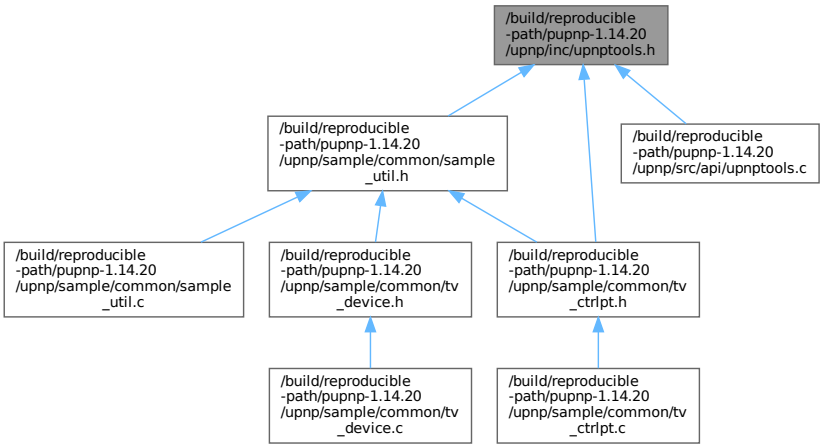
9.61 /build/reproducible-path/pupnp-1.14.20/upnp/inc/upnptools.h File Reference

```
#include "ixml.h"
#include "upnpconfig.h"
```

Include dependency graph for upnptools.h:



This graph shows which files directly or indirectly include this file:



Functions

- UPNP_EXPORT_SPEC void UpnpSetMaxJobsTotal (int mjt)

  Sets the maximum number of jobs in the internal thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.
- UPNP_EXPORT_SPEC const char * UpnpGetErrorMessage (int errorcode)

  Converts an SDK error code into a string error message suitable for display. The memory returned from this function should NOT be freed.
- UPNP_EXPORT_SPEC int UpnpResolveURL (const char *BaseURL, const char *RelURL, char *AbsURL)

  Combines a base URL and a relative URL into a single absolute URL.
- UPNP_EXPORT_SPEC int UpnpResolveURL2 (const char *BaseURL, const char *RelURL, char **Abs↩ URL)

  Combines a base URL and a relative URL into a single absolute URL.
- UPNP_EXPORT_SPEC IXML_Document * UpnpMakeAction (const char *ActionName, const char *Serv↩ Type, int NumArg, const char *Arg,...)

  Creates an action request packet based on its input parameters (status variable name and value pair).
- UPNP_EXPORT_SPEC IXML_Document * UpnpMakeActionResponse (const char *ActionName, const char *ServType, int NumArg, const char *Arg,...)

  Ceates an action response packet based on its output parameters (status variable name and value pair).
- UPNP_EXPORT_SPEC int UpnpAddToAction (IXML_Document **ActionDoc, const char *ActionName, const char *ServType, const char *ArgName, const char *ArgVal)

  Adds the argument in the action request.
- UPNP_EXPORT_SPEC int UpnpAddToActionResponse (IXML_Document **ActionResponse, const char *ActionName, const char *ServType, const char *ArgName, const char *ArgVal)

  Creates an action response packet based on its output parameters (status variable name and value pair).
- UPNP_EXPORT_SPEC IXML_Document * UpnpCreatePropertySet (int NumArg, const char *Arg,...)

  Creates a property set message packet.
- UPNP_EXPORT_SPEC int UpnpAddToPropertySet (IXML_Document **PropSet, const char *ArgName, const char *ArgVal)

  Can be used when an application needs to transfer the status of many variables at once.

9.62 upnptools.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * * Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * * Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * * Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef UPNP_TOOLS_H
00033 #define UPNP_TOOLS_H
00034
00049 #include "ixml.h" /* for IXML_Document */
00050 #include "upnpconfig.h" /* for UPNP_HAVE_TOOLS */
00051
00052 /* Function declarations only if tools compiled into the library */
00053 #if UPNP_HAVE_TOOLS
00054
00055         #ifdef __cplusplus
00056 extern "C" {
00057         #endif
00058
00064 UPNP_EXPORT_SPEC void UpnpSetMaxJobsTotal(int mjt);
00065
00073 UPNP_EXPORT_SPEC const char *UpnpGetErrorMessage(
00075         int errorcode);
00076
00091 UPNP_EXPORT_SPEC int UpnpResolveURL(
00093         const char *BaseURL,
00095         const char *RelURL,
00097         char *AbsURL);
00098
00113 UPNP_EXPORT_SPEC int UpnpResolveURL2(
00115         const char *BaseURL,
00117         const char *RelURL,
00120         char **AbsURL);
00121
00135 UPNP_EXPORT_SPEC IXML_Document *UpnpMakeAction(
00137         const char *ActionName,
00139         const char *ServType,
00141         int NumArg,
00143         const char *Arg,
00145         ...);
00146
00160 UPNP_EXPORT_SPEC IXML_Document *UpnpMakeActionResponse(
00162         const char *ActionName,
00164         const char *ServType,
00166         int NumArg,
00168         const char *Arg,
00170         ...);
00171
00189 UPNP_EXPORT_SPEC int UpnpAddToAction(
00191         IXML_Document **ActionDoc,
00193         const char *ActionName,
00195         const char *ServType,
00197         const char *ArgName,
00199         const char *ArgVal);
00200
00219 UPNP_EXPORT_SPEC int UpnpAddToActionResponse(
00221         IXML_Document **ActionResponse,
```

```
00223         const char *ActionName,
00225         const char *ServType,
00227         const char *ArgName,
00229         const char *ArgVal);
00230
00239 UPNP_EXPORT_SPEC IXML_Document *UpnpCreatePropertySet(
00241         int NumArg,
00243         const char *Arg,
00245         ...);
00246
00262 UPNP_EXPORT_SPEC int UpnpAddToPropertySet(
00265         IXML_Document **PropSet,
00267         const char *ArgName,
00269         const char *ArgVal);
00270
00271         #ifdef __cplusplus
00272 }
00273         #endif
00274
00277 #endif /* UPNP_HAVE_TOOLS */
00278
00279 #endif /* UPNP_TOOLS_H */
```

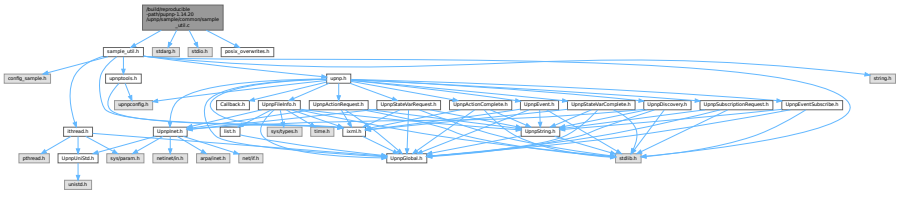
9.63 UpnpUniStd.h

```
00001 #ifndef UPNPUNISTD_H
00002 #define UPNPUNISTD_H
00003
00004 #ifdef _WIN32
00005         /* Do not #include <unistd.h> on WIN32. */
00006 #else                           /* _WIN32 */
00007         #include <unistd.h> /* for close() */
00008 #endif                          /* _WIN32 */
00009
00010 #endif /* UPNPUNISTD_H */
```

9.64 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util.c File Reference

```
#include "sample_util.h"
#include <stdarg.h>
#include <stdio.h>
#include "posix_overwrites.h"
```

Include dependency graph for sample_util.c:



Macros

- #define **SAMPLE_UTIL_C**
- #define **OLD_FIND_SERVICE_CODE**
- #define **MAX_BUF** (8 * 1024)

Functions

- int SampleUtil_Initialize (print_string print_function)

  Initializes the sample util. Must be called before any sample util functions. May be called multiple times.
- int SampleUtil_RegisterUpdateFunction (state_update update_function)
- int SampleUtil_Finish ()

  Releases Resources held by sample util.
- char * SampleUtil_GetElementValue (IXML_Element *element)

  Given a DOM node such as <Channel>11</Channel>, this routine extracts the value (e.g., 11) from the node and returns it as a string. The string must be freed by the caller using free.
- IXML_NodeList * SampleUtil_GetFirstServiceList (IXML_Document *doc)

  Given a DOM node representing a UPnP Device Description Document, this routine parses the document and finds the first service list (i.e., the service list for the root device). The service list is returned as a DOM node list. The NodeList must be freed using NodeList_free.
- char * SampleUtil_GetFirstDocumentItem (IXML_Document *doc, const char *item)

  Given a document node, this routine searches for the first element named by the input string item, and returns its value as a string. String must be freed by caller using free.
- char * SampleUtil_GetFirstElementItem (IXML_Element *element, const char *item)

  Given a DOM element, this routine searches for the first element named by the input string item, and returns its value as a string. The string must be freed using free.
- void SampleUtil_PrintEventType (Upnp_EventType S)

  Prints a callback event type as a string.
- int SampleUtil_PrintEvent (Upnp_EventType EventType, const void *Event)

  Prints callback event structure details.
- int SampleUtil_FindAndParseService (IXML_Document *DescDoc, const char *location, const char *serviceType, char **serviceId, char **eventURL, char **controlURL)

  This routine finds the first occurance of a service in a DOM representation of a description document and parses it. Note that this function currently assumes that the eventURL and controlURL values in the service definitions are full URLs. Relative URLs are not handled here.
- int SampleUtil_Print (const char *fmt, ...)

  Function emulating printf that ultimately calls the registered print function with the formatted string.
- void SampleUtil_StateUpdate (const char *varName, const char *varValue, const char *UDN, eventType type)
- void linux_print (const char *format,...)

  Prints a string to standard out.

Variables

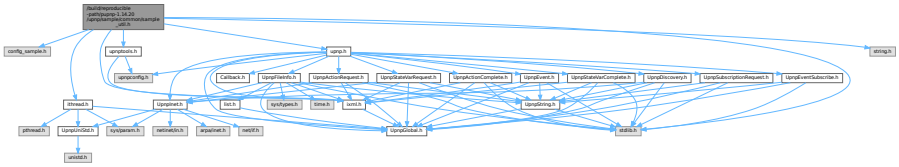
- static int **initialize_init** = 1
- static int **initialize_register** = 1
- print_string gPrintFun = NULL
- state_update gStateUpdateFun = NULL
- ithread_mutex_t display_mutex

9.65 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util.h File Reference

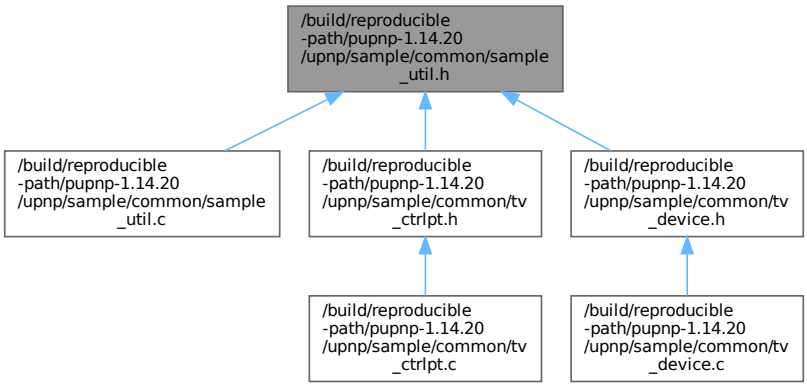
```
#include "config_sample.h"
#include "ithread.h"
#include "ixml.h"
#include "upnp.h"
```

```
#include "upnptools.h"
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for sample_util.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef void(∗ print_string) (const char ∗string,...)

  Prototype for displaying strings. All printing done by the device, control point, and sample util, ultimately use this to display strings to the user.
- typedef void(∗ state_update) (const char ∗varName, const char ∗varValue, const char ∗UDN, eventType type)

  Prototype for passing back state changes.

Enumerations

- enum **eventType** { **STATE_UPDATE** = 0 , **DEVICE_ADDED** = 1 , **DEVICE_REMOVED** = 2 , **GET_VAR_↩ COMPLETE** = 3 }

Functions

- char ∗ SampleUtil_GetElementValue (IXML_Element ∗element)

  Given a DOM node such as <Channel>11</Channel>, this routine extracts the value (e.g., 11) from the node and returns it as a string. The string must be freed by the caller using free.
- IXML_NodeList ∗ SampleUtil_GetFirstServiceList (IXML_Document ∗doc)

Given a DOM node representing a UPnP Device Description Document, this routine parses the document and finds the first service list (i.e., the service list for the root device). The service list is returned as a DOM node list. The NodeList must be freed using NodeList_free.

- char * SampleUtil_GetFirstDocumentItem (IXML_Document *doc, const char *item)

  Given a document node, this routine searches for the first element named by the input string item, and returns its value as a string. String must be freed by caller using free.
- char * SampleUtil_GetFirstElementItem (IXML_Element *element, const char *item)

  Given a DOM element, this routine searches for the first element named by the input string item, and returns its value as a string. The string must be freed using free.
- void SampleUtil_PrintEventType (Upnp_EventType S)

  Prints a callback event type as a string.
- int SampleUtil_PrintEvent (Upnp_EventType EventType, const void *Event)

  Prints callback event structure details.
- int SampleUtil_FindAndParseService (IXML_Document *DescDoc, const char *location, const char *serviceType, char **serviceId, char **eventURL, char **controlURL)

  This routine finds the first occurance of a service in a DOM representation of a description document and parses it. Note that this function currently assumes that the eventURL and controlURL values in the service definitions are full URLs. Relative URLs are not handled here.
- int SampleUtil_Initialize (print_string print_function)

  Initializes the sample util. Must be called before any sample util functions. May be called multiple times.
- int SampleUtil_Finish ()

  Releases Resources held by sample util.
- int SampleUtil_Print (const char *fmt, ...)

  Function emulating printf that ultimately calls the registered print function with the formatted string.
- int SampleUtil_RegisterUpdateFunction (state_update update_function)
- void SampleUtil_StateUpdate (const char *varName, const char *varValue, const char *UDN, eventType type)
- void linux_print (const char *format,...)

  Prints a string to standard out.

Variables

- const char * **TvServiceType** [ ]
- ithread_mutex_t display_mutex
- print_string gPrintFun
- state_update gStateUpdateFun

9.66 sample_util.h

Go to the documentation of this file.

```
00001 #ifndef SAMPLE_UTIL_H
00002 #define SAMPLE_UTIL_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  *
00009  * Redistribution and use in source and binary forms, with or without
00010  * modification, are permitted provided that the following conditions are met:
00011  *
00012  * - Redistributions of source code must retain the above copyright notice,
00013  * this list of conditions and the following disclaimer.
00014  * - Redistributions in binary form must reproduce the above copyright notice,
00015  * this list of conditions and the following disclaimer in the documentation
00016  * and/or other materials provided with the distribution.
00017  * - Neither name of Intel Corporation nor the names of its contributors
00018  * may be used to endorse or promote products derived from this software
```

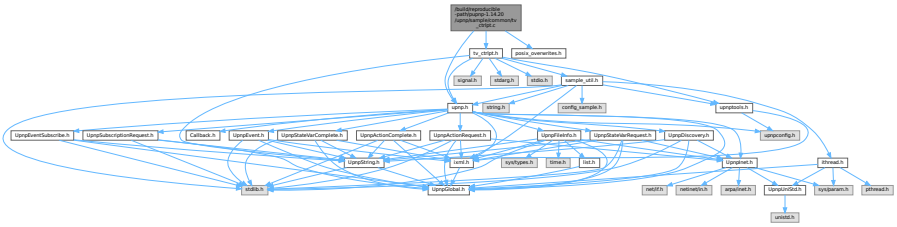
```
00019  * without specific prior written permission.
00020  *
00021  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00022  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00023  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00024  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00025  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00026  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00027  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00028  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00029  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00030  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00031  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00032  *
00033  ******************************************************************************/
00034
00043 #ifdef __cplusplus
00044 extern "C" {
00045 #endif /* __cplusplus */
00046
00047 #include "config_sample.h"
00048 #include "ithread.h"
00049 #include "ixml.h" /* for IXML_Document, IXML_Element */
00050 #include "upnp.h" /* for Upnp_EventType */
00051 #include "upnptools.h"
00052
00053 #include <stdlib.h>
00054 #include <string.h>
00055
00056 #ifdef SAMPLE_UTIL_C
00058 const char *TvServiceType[] = {"urn:schemas-upnp-org:service:tvcontrol:1",
00059         "urn:schemas-upnp-org:service:tvpicture:1"};
00060 #else  /* SAMPLE_UTIL_C */
00061 extern const char *TvServiceType[];
00062 #endif /* SAMPLE_UTIL_C */
00063
00064 /* mutex to control displaying of events */
00065 extern ithread_mutex_t display_mutex;
00066
00067 typedef enum
00068 {
00069         STATE_UPDATE = 0,
00070         DEVICE_ADDED = 1,
00071         DEVICE_REMOVED = 2,
00072         GET_VAR_COMPLETE = 3
00073 } eventType;
00074
00082 char *SampleUtil_GetElementValue(
00084         IXML_Element *element);
00085
00095 IXML_NodeList *SampleUtil_GetFirstServiceList(
00097         IXML_Document *doc);
00098
00104 char *SampleUtil_GetFirstDocumentItem(
00106         IXML_Document *doc,
00108         const char *item);
00109
00115 char *SampleUtil_GetFirstElementItem(
00117         IXML_Element *element,
00119         const char *item);
00120
00124 void SampleUtil_PrintEventType(
00126         Upnp_EventType S);
00127
00131 int SampleUtil_PrintEvent(
00133         Upnp_EventType EventType,
00135         const void *Event);
00136
00143 int SampleUtil_FindAndParseService(
00145         IXML_Document *DescDoc,
00147         const char *location,
00149         const char *serviceType,
00151         char **serviceId,
00153         char **eventURL,
00155         char **controlURL);
00156
00162 typedef void (*print_string)(
00164         const char *string,
00166         ...)
00167 #if (__GNUC__ >= 3)
00168         /* This enables printf like format checking by the compiler */
00169         __attribute__((format(__printf__, 1, 2)))
00170 #endif
00171         ;
00172
00174 extern print_string gPrintFun;
00175
```

```
00179 typedef void (*state_update)(
00181         const char *varName,
00183         const char *varValue,
00185         const char *UDN,
00187         eventType type);
00188
00190 extern state_update gStateUpdateFun;
00191
00196 int SampleUtil_Initialize(
00198         print_string print_function);
00199
00203 int SampleUtil_Finish();
00204
00216 int SampleUtil_Print(
00218         const char *fmt,
00220         ...)
00221 #if (__GNUC__ >= 3)
00222         /* This enables printf like format checking by the compiler */
00223         __attribute__((format(__printf__, 1, 2)))
00224 #endif
00225         ;
00226
00230 int SampleUtil_RegisterUpdateFunction(
00232         state_update update_function);
00233
00237 void SampleUtil_StateUpdate(
00239         const char *varName,
00241         const char *varValue,
00243         const char *UDN,
00245         eventType type);
00246
00250 void linux_print(const char *format, ...)
00251 #if (__GNUC__ >= 3)
00252         /* This enables printf like format checking by the compiler */
00253         __attribute__((format(__printf__, 1, 2)))
00254 #endif
00255         ;
00256
00257 #ifdef __cplusplus
00258 };
00259 #endif /* __cplusplus */
00260
00263 #endif /* SAMPLE_UTIL_H */
```

9.67 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.c File Reference

```
#include "tv_ctrlpt.h"
#include "upnp.h"
#include "posix_overwrites.h"
```

Include dependency graph for tv_ctrlpt.c:



Data Structures

- struct cmdloop_commands

Control Point Sample Module

- enum cmdloop_tvcmds {
  PRTHELP = 0 , **PRTFULLHELP** , **POWON** , **POWOFF** ,
  SETCHAN , **SETVOL** , **SETCOL** , **SETTINT** ,
  SETCONT , **SETBRT** , **CTRLACTION** , **PICTACTION** ,
  CTRLGETVAR , **PICTGETVAR** , **PRTDEV** , **LSTDEV** ,
  REFRESH , **EXITCMD** }
- ithread_mutex_t DeviceListMutex
- UpnpClient_Handle **ctrlpt_handle** = -1
- const char TvDeviceType [ ] = "urn:schemas-upnp-org:device:tvdevice:1"
- const char * TvServiceName [ ] = {"Control", "Picture"}
- const char * TvVarName [2][4]
- char TvVarCount [2]
- int default_timeout = 1801
- struct TvDeviceNode * GlobalDeviceList = NULL
- static int TvCtrlPointTimerLoopRun = 1

  Function that runs in its own thread and monitors advertisement and subscription timeouts for devices in the global device list.
- struct cmdloop_commands **cmdloop_commands**
- static struct cmdloop_commands cmdloop_cmdlist [ ]
- int **TvCtrlPointDeleteNode** (struct TvDeviceNode *node)
- int **TvCtrlPointRemoveDevice** (const char *UDN)
- int **TvCtrlPointRemoveAll** (void)
- int **TvCtrlPointRefresh** (void)
- int **TvCtrlPointGetVar** (int service, int devnum, const char *varname)
- int **TvCtrlPointGetPower** (int devnum)
- int **TvCtrlPointGetChannel** (int devnum)
- int **TvCtrlPointGetVolume** (int devnum)
- int **TvCtrlPointGetColor** (int devnum)
- int **TvCtrlPointGetTint** (int devnum)
- int **TvCtrlPointGetContrast** (int devnum)
- int **TvCtrlPointGetBrightness** (int devnum)
- int **TvCtrlPointSendAction** (int service, int devnum, const char *actionname, const char **param_name, char **param_val, int param_count)
- int **TvCtrlPointSendActionNumericArg** (int devnum, int service, const char *actionName, const char *paramName, int paramValue)
- int **TvCtrlPointSendPowerOn** (int devnum)
- int **TvCtrlPointSendPowerOff** (int devnum)
- int **TvCtrlPointSendSetChannel** (int devnum, int channel)
- int **TvCtrlPointSendSetVolume** (int devnum, int volume)
- int **TvCtrlPointSendSetColor** (int devnum, int color)
- int **TvCtrlPointSendSetTint** (int devnum, int tint)
- int **TvCtrlPointSendSetContrast** (int devnum, int contrast)
- int **TvCtrlPointSendSetBrightness** (int devnum, int brightness)
- int **TvCtrlPointGetDevice** (int devnum, struct TvDeviceNode **devnode)
- int **TvCtrlPointPrintList** ()
- int **TvCtrlPointPrintDevice** (int devnum)
- void **TvCtrlPointAddDevice** (IXML_Document *DescDoc, const char *location, int expires)
- void TvStateUpdate (char *UDN, int Service, IXML_Document *ChangedVariables, char **State)

  Update a Tv state table. Called when an event is received.
- void **TvCtrlPointHandleEvent** (const char *sid, int evntkey, IXML_Document *changes)
- void **TvCtrlPointHandleSubscribeUpdate** (const char *eventURL, const Upnp_SID sid, int timeout)
- void **TvCtrlPointHandleGetVar** (const char *controlURL, const char *varName, const DOMString varValue)
- int **TvCtrlPointCallbackEventHandler** (Upnp_EventType EventType, const void *Event, void *Cookie)

- void TvCtrlPointVerifyTimeouts (int incr)

  Checks the advertisement each device in the global device list.
- void * **TvCtrlPointTimerLoop** (void *args)
- int TvCtrlPointStart (char *iface, state_update updateFunctionPtr, int combo)

  Call this function to initialize the UPnP library and start the TV Control Point. This function creates a timer thread and provides a callback handler to process any UPnP events that are received.
- int **TvCtrlPointStop** (void)
- void TvCtrlPointPrintShortHelp (void)

  Print help info for this application.
- void TvCtrlPointPrintLongHelp (void)

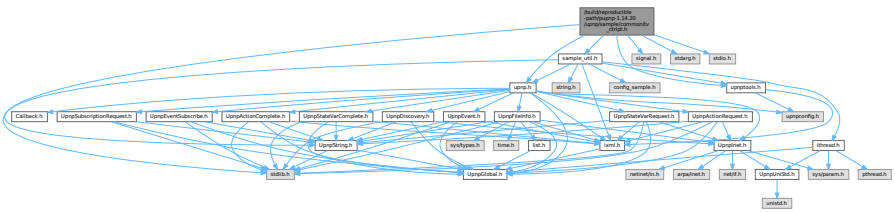
  Print long help info for this application.
- void TvCtrlPointPrintCommands (void)
- void * TvCtrlPointCommandLoop (void *args)

  Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands.
- int **TvCtrlPointProcessCommand** (char *cmdline)

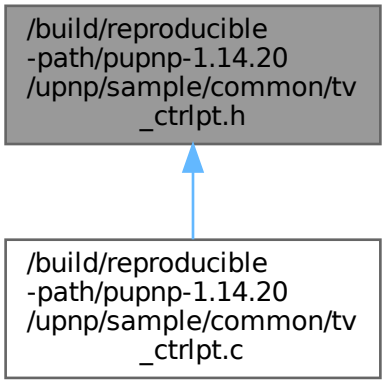
9.68 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_↩ ctrlpt.h File Reference

```
#include "sample_util.h"
#include "UpnpString.h"
#include "upnp.h"
#include "upnptools.h"
#include <signal.h>
#include <stdarg.h>
#include <stdio.h>
```

Include dependency graph for tv_ctrlpt.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct tv_service
- struct TvDevice
- struct TvDeviceNode

Macros

Device Sample API

- #define TV_SERVICE_SERVCOUNT 2
- #define TV_SERVICE_CONTROL 0
- #define TV_SERVICE_PICTURE 1
- #define TV_CONTROL_VARCOUNT 3
- #define TV_CONTROL_POWER 0
- #define TV_CONTROL_CHANNEL 1
- #define TV_CONTROL_VOLUME 2
- #define TV_PICTURE_VARCOUNT 4
- #define TV_PICTURE_COLOR 0
- #define TV_PICTURE_TINT 1
- #define TV_PICTURE_CONTRAST 2
- #define TV_PICTURE_BRIGHTNESS 3
- #define TV_MAX_VAL_LEN 5
- #define TV_MAXVARS TV_PICTURE_VARCOUNT

Functions

Control Point Sample Module

- int **TvCtrlPointDeleteNode** (struct TvDeviceNode *node)
- int **TvCtrlPointRemoveDevice** (const char *UDN)
- int **TvCtrlPointRemoveAll** (void)
- int **TvCtrlPointRefresh** (void)

- int **TvCtrlPointSendAction** (int service, int devnum, const char *actionname, const char **param_name, char **param_val, int param_count)
- int **TvCtrlPointSendActionNumericArg** (int devnum, int service, const char *actionName, const char *paramName, int paramValue)
- int **TvCtrlPointSendPowerOn** (int devnum)
- int **TvCtrlPointSendPowerOff** (int devnum)
- int **TvCtrlPointSendSetChannel** (int devnum, int channel)
- int **TvCtrlPointSendSetVolume** (int devnum, int volume)
- int **TvCtrlPointSendSetColor** (int devnum, int color)
- int **TvCtrlPointSendSetTint** (int devnum, int tint)
- int **TvCtrlPointSendSetContrast** (int devnum, int contrast)
- int **TvCtrlPointSendSetBrightness** (int devnum, int brightness)
- int **TvCtrlPointGetVar** (int service, int devnum, const char *varname)
- int **TvCtrlPointGetPower** (int devnum)
- int **TvCtrlPointGetChannel** (int devnum)
- int **TvCtrlPointGetVolume** (int devnum)
- int **TvCtrlPointGetColor** (int devnum)
- int **TvCtrlPointGetTint** (int devnum)
- int **TvCtrlPointGetContrast** (int devnum)
- int **TvCtrlPointGetBrightness** (int devnum)
- int **TvCtrlPointGetDevice** (int devnum, struct TvDeviceNode **devnode)
- int **TvCtrlPointPrintList** ()
- int **TvCtrlPointPrintDevice** (int devnum)
- void **TvCtrlPointAddDevice** (IXML_Document *DescDoc, const char *location, int expires)
- void **TvCtrlPointHandleGetVar** (const char *controlURL, const char *varName, const DOMString varValue)
- void TvStateUpdate (char *UDN, int Service, IXML_Document *ChangedVariables, char **State)

  Update a Tv state table. Called when an event is received.
- void **TvCtrlPointHandleEvent** (const char *sid, int evntkey, IXML_Document *changes)
- void **TvCtrlPointHandleSubscribeUpdate** (const char *eventURL, const Upnp_SID sid, int timeout)
- int **TvCtrlPointCallbackEventHandler** (Upnp_EventType EventType, const void *Event, void *Cookie)
- void TvCtrlPointVerifyTimeouts (int incr)

  Checks the advertisement each device in the global device list.
- void TvCtrlPointPrintCommands (void)
- void * TvCtrlPointCommandLoop (void *args)

  Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands.
- int TvCtrlPointStart (char *iface, state_update updateFunctionPtr, int combo)

  Call this function to initialize the UPnP library and start the TV Control Point. This function creates a timer thread and provides a callback handler to process any UPnP events that are received.
- int **TvCtrlPointStop** (void)
- int **TvCtrlPointProcessCommand** (char *cmdline)
- void TvCtrlPointPrintShortHelp (void)

  Print help info for this application.
- void TvCtrlPointPrintLongHelp (void)

  Print long help info for this application.

Contro Point Sample API

- #define **TV_SUCCESS** 0
- #define **TV_ERROR** (-1)
- #define **TV_WARNING** 1
- const char * TvServiceName [ ]
- const char * TvVarName [2][4]
- char **TvVarCount** [ ]
- struct TvDeviceNode * GlobalDeviceList
- ithread_mutex_t DeviceListMutex
- UpnpClient_Handle **ctrlpt_handle**
- void **TvCtrlPointPrintHelp** (void)

9.69 tv_ctrlpt.h

Go to the documentation of this file.

```
00001 #ifndef UPNP_TV_CTRLPT_H
00002 #define UPNP_TV_CTRLPT_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  *
00009  * Redistribution and use in source and binary forms, with or without
00010  * modification, are permitted provided that the following conditions are met:
00011  *
00012  * - Redistributions of source code must retain the above copyright notice,
00013  * this list of conditions and the following disclaimer.
00014  * - Redistributions in binary form must reproduce the above copyright notice,
00015  * this list of conditions and the following disclaimer in the documentation
00016  * and/or other materials provided with the distribution.
00017  * - Neither name of Intel Corporation nor the names of its contributors
00018  * may be used to endorse or promote products derived from this software
00019  * without specific prior written permission.
00020  *
00021  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00022  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00023  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00024  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00025  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00026  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00027  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00028  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00029  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00030  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00031  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00032  *
00033  ******************************************************************************/
00034
00047 #ifdef __cplusplus
00048 extern "C" {
00049 #endif
00050
00051 #include "sample_util.h"
00052
00053 #include "UpnpString.h"
00054 #include "upnp.h"
00055 #include "upnptools.h"
00056
00057 #include <signal.h>
00058 #include <stdarg.h>
00059 #include <stdio.h>
00060
00061 #define TV_SERVICE_SERVCOUNT 2
00062 #define TV_SERVICE_CONTROL 0
00063 #define TV_SERVICE_PICTURE 1
00064
00065 #define TV_CONTROL_VARCOUNT 3
00066 #define TV_CONTROL_POWER 0
00067 #define TV_CONTROL_CHANNEL 1
00068 #define TV_CONTROL_VOLUME 2
00069
00070 #define TV_PICTURE_VARCOUNT 4
00071 #define TV_PICTURE_COLOR 0
00072 #define TV_PICTURE_TINT 1
00073 #define TV_PICTURE_CONTRAST 2
00074 #define TV_PICTURE_BRIGHTNESS 3
00075
00076 #define TV_MAX_VAL_LEN 5
00077
00078 #define TV_SUCCESS 0
00079 #define TV_ERROR (-1)
00080 #define TV_WARNING 1
00081
00082 /* This should be the maximum VARCOUNT from above */
00083 #define TV_MAXVARS TV_PICTURE_VARCOUNT
00084
00085 extern const char *TvServiceName[];
00086 extern const char *TvVarName[TV_SERVICE_SERVCOUNT][TV_MAXVARS];
00087 extern char TvVarCount[];
00088
00089 struct tv_service
00090 {
00091         char ServiceId[NAME_SIZE];
00092         char ServiceType[NAME_SIZE];
00093         char *VariableStrVal[TV_MAXVARS];
00094         char EventURL[NAME_SIZE];
```

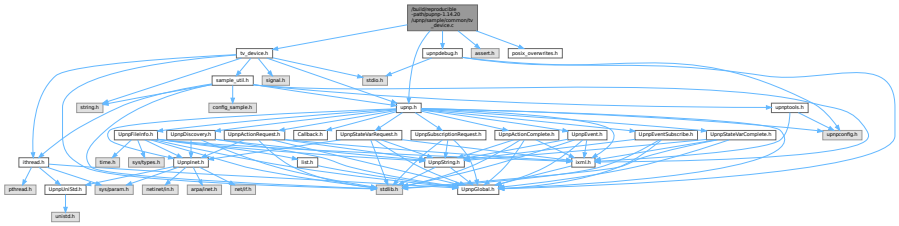
```
00095         char ControlURL[NAME_SIZE];
00096         char SID[NAME_SIZE];
00097 };
00098
00099 extern struct TvDeviceNode *GlobalDeviceList;
00100
00101 struct TvDevice
00102 {
00103         char UDN[250];
00104         char DescDocURL[250];
00105         char FriendlyName[250];
00106         char PresURL[250];
00107         int AdvrTimeOut;
00108         struct tv_service TvService[TV_SERVICE_SERVCOUNT];
00109 };
00110
00111 struct TvDeviceNode
00112 {
00113         struct TvDevice device;
00114         struct TvDeviceNode *next;
00115 };
00116
00117 extern ithread_mutex_t DeviceListMutex;
00118
00119 extern UpnpClient_Handle ctrlpt_handle;
00120
00121 void TvCtrlPointPrintHelp(void);
00122 int TvCtrlPointDeleteNode(struct TvDeviceNode *);
00123 int TvCtrlPointRemoveDevice(const char *);
00124 int TvCtrlPointRemoveAll(void);
00125 int TvCtrlPointRefresh(void);
00126
00127 int TvCtrlPointSendAction(int, int, const char *, const char **, char **, int);
00128 int TvCtrlPointSendActionNumericArg(int devnum,
00129         int service,
00130         const char *actionName,
00131         const char *paramName,
00132         int paramValue);
00133 int TvCtrlPointSendPowerOn(int devnum);
00134 int TvCtrlPointSendPowerOff(int devnum);
00135 int TvCtrlPointSendSetChannel(int, int);
00136 int TvCtrlPointSendSetVolume(int, int);
00137 int TvCtrlPointSendSetColor(int, int);
00138 int TvCtrlPointSendSetTint(int, int);
00139 int TvCtrlPointSendSetContrast(int, int);
00140 int TvCtrlPointSendSetBrightness(int, int);
00141
00142 int TvCtrlPointGetVar(int, int, const char *);
00143 int TvCtrlPointGetPower(int devnum);
00144 int TvCtrlPointGetChannel(int);
00145 int TvCtrlPointGetVolume(int);
00146 int TvCtrlPointGetColor(int);
00147 int TvCtrlPointGetTint(int);
00148 int TvCtrlPointGetContrast(int);
00149 int TvCtrlPointGetBrightness(int);
00150
00151 int TvCtrlPointGetDevice(int, struct TvDeviceNode **);
00152 int TvCtrlPointPrintList(void);
00153 int TvCtrlPointPrintDevice(int);
00154 void TvCtrlPointAddDevice(IXML_Document *, const char *, int);
00155 void TvCtrlPointHandleGetVar(const char *, const char *, const DOMString);
00156
00163 void TvStateUpdate(
00165         char *UDN,
00167         int Service,
00169         IXML_Document *ChangedVariables,
00171         char **State);
00172
00173 void TvCtrlPointHandleEvent(const char *, int, IXML_Document *);
00174 void TvCtrlPointHandleSubscribeUpdate(const char *, const Upnp_SID, int);
00175 int TvCtrlPointCallbackEventHandler(Upnp_EventType, const void *, void *);
00176
00185 void TvCtrlPointVerifyTimeouts(
00188         int incr);
00189
00190 void TvCtrlPointPrintCommands(void);
00191 void *TvCtrlPointCommandLoop(void *);
00192 int TvCtrlPointStart(char *iface, state_update updateFunctionPtr, int combo);
00193 int TvCtrlPointStop(void);
00194 int TvCtrlPointProcessCommand(char *cmdline);
00195
00199 void TvCtrlPointPrintShortHelp(void);
00200
00204 void TvCtrlPointPrintLongHelp(void);
00205
00209 void TvCtrlPointPrintCommands(void);
00210
```

```
00216 void *TvCtrlPointCommandLoop(void *args);
00217
00221 int TvCtrlPointProcessCommand(char *cmdline);
00222
00223 #ifdef __cplusplus
00224 };
00225 #endif
00226
00231 #endif /* UPNP_TV_CTRLPT_H */
```

9.70 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.c File Reference

```
#include "tv_device.h"
#include "upnp.h"
#include "upnpdebug.h"
#include <assert.h>
#include "posix_overwrites.h"
```

Include dependency graph for tv_device.c:



Macros

Device Sample API

- #define MAX_COLOR 10
- #define **MIN_COLOR** 1
- #define MAX_BRIGHTNESS 10
- #define **MIN_BRIGHTNESS** 1
- #define POWER_ON 1
- #define **POWER_OFF** 0
- #define MAX_TINT 10
- #define **MIN_TINT** 1
- #define MAX_VOLUME 10
- #define **MIN_VOLUME** 1
- #define MAX_CONTRAST 10
- #define **MIN_CONTRAST** 1
- #define MAX_CHANNEL 100
- #define **MIN_CHANNEL** 1

Device Sample Module

- #define **DEFAULT_WEB_DIR** "./web"
- #define **DESC_URL_SIZE** 200
- const char * tvc_varname [ ] = {"Power", "Channel", "Volume"}
- char **tvc_varval** [3][5]
- const char * **tvc_varval_def** [ ] = {"1", "1", "5"}

- const char * tvp_varname [ ] = {"Color", "Tint", "Contrast", "Brightness"}
- char **tvp_varval** [4][5]
- const char * **tvp_varval_def** [ ] = {"5", "5", "5", "5"}
- int default_advr_expire = 100
- struct TvService tv_service_table [2]
- UpnpDevice_Handle device_handle = -1
- ithread_mutex_t TVDevMutex
- static int SetServiceTable (int serviceType, const char *UDN, const char *serviceId, const char *service↩ TypeS, struct TvService *out)

  Initializes the service table for the specified service.
- int SetActionTable (int serviceType, struct TvService *out)

  Initializes the action table for the specified service.
- int TvDeviceStateTableInit (char *DescDocURL)

  Initialize the device state table for this TvDevice, pulling identifier info from the description Document.
- int TvDeviceHandleSubscriptionRequest (const UpnpSubscriptionRequest *sr_event)

  Called during a subscription request callback.
- int TvDeviceHandleGetVarRequest (UpnpStateVarRequest *cgv_event)

  Called during a get variable request callback.
- int TvDeviceHandleActionRequest (UpnpActionRequest *ca_event)

  Called during an action request callback.
- int TvDeviceSetServiceTableVar (unsigned int service, int variable, char *value)

  Update the TvDevice service state table, and notify all subscribed control points of the updated state.
- static int TvDeviceSetPower (int on)

  Turn the power on/off, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- int TvDevicePowerOn (IXML_Document *in, IXML_Document **out, const char **errorString)

  Turn the power on.
- int TvDevicePowerOff (IXML_Document *in, IXML_Document **out, const char **errorString)

  Turn the power off.
- int TvDeviceSetChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the channel, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- int **IncrementChannel** (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)
- int TvDeviceDecreaseChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the channel.
- int TvDeviceIncreaseChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the channel.
- int TvDeviceSetVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the volume, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- static int IncrementVolume (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)

  Increment the volume. Read the current volume from the state table, add the increment, and then change the volume.
- int TvDeviceIncreaseVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the volume.
- int TvDeviceDecreaseVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the volume.
- int TvDeviceSetColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the color, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- static int IncrementColor (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)

  Increment the color. Read the current color from the state table, add the increment, and then change the color.
- int TvDeviceDecreaseColor (IXML_Document *in, IXML_Document **out, const char **errorString)

Decrease the color.

- int TvDeviceIncreaseColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the color.

- int TvDeviceSetTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the tint, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.

- int **IncrementTint** (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)
- int TvDeviceIncreaseTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase tint.

- int TvDeviceDecreaseTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease tint.

- int TvDeviceSetContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the contrast, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.

- static int IncrementContrast (int incr, IXML_Document *in, IXML_Document **out, const char **errorString)

  Increment the contrast. Read the current contrast from the state table, add the increment, and then change the contrast.

- int TvDeviceIncreaseContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the contrast.

- int TvDeviceDecreaseContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the contrast.

- int TvDeviceSetBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the brightness, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.

- static int IncrementBrightness (int incr, IXML_Document *in, IXML_Document **out, const char **error↩ String)

  Increment the brightness. Read the current brightness from the state table, add the increment, and then change the brightness.

- int TvDeviceIncreaseBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase brightnesss.

- int TvDeviceDecreaseBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease brightnesss.

- int TvDeviceCallbackEventHandler (Upnp_EventType, const void *Event, void *Cookie)

  The callback handler registered with the SDK while registering root device.

- int TvDeviceStart (char *iface, unsigned short port, const char *desc_doc_name, const char *web_dir_path, int ip_mode, print_string pfun, int combo)

  Initializes the UPnP Sdk, registers the device, and sends out advertisements.

- int TvDeviceStop (void)

  Stops the device. Uninitializes the sdk.

- void * TvDeviceCommandLoop (void *args)

  Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands. Only one command, exit, is currently defined.

- int device_main (int argc, char *argv[])

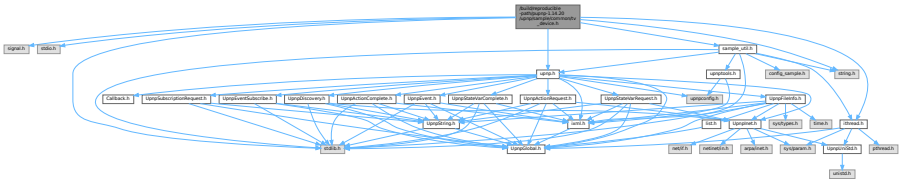
  Main entry point for tv device application.

9.71 /build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_↩ device.h File Reference

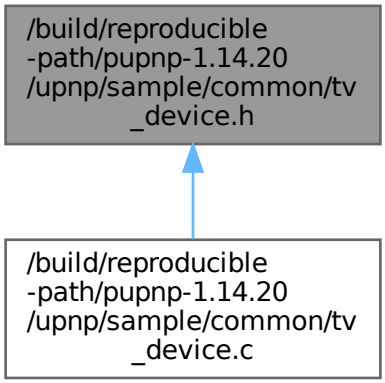
```
#include <signal.h>
#include <stdio.h>
```

```
#include "sample_util.h"
#include "ithread.h"
#include "upnp.h"
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for tv_device.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct TvService

Functions

Device Sample Module

- int SetActionTable (int serviceType, struct TvService *out)

  Initializes the action table for the specified service.
- int TvDeviceStateTableInit (char *DescDocURL)

  Initialize the device state table for this TvDevice, pulling identifier info from the description Document.
- int TvDeviceHandleSubscriptionRequest (const UpnpSubscriptionRequest *sr_event)

  Called during a subscription request callback.
- int TvDeviceHandleGetVarRequest (UpnpStateVarRequest *cgv_event)

  Called during a get variable request callback.
- int TvDeviceHandleActionRequest (UpnpActionRequest *ca_event)

Called during an action request callback.

- int TvDeviceCallbackEventHandler (Upnp_EventType, const void *Event, void *Cookie)

  The callback handler registered with the SDK while registering root device.
- int TvDeviceSetServiceTableVar (unsigned int service, int variable, char *value)

  Update the TvDevice service state table, and notify all subscribed control points of the updated state.
- int TvDevicePowerOn (IXML_Document *in, IXML_Document **out, const char **errorString)

  Turn the power on.
- int TvDevicePowerOff (IXML_Document *in, IXML_Document **out, const char **errorString)

  Turn the power off.
- int TvDeviceSetChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the channel, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- int TvDeviceIncreaseChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the channel.
- int TvDeviceDecreaseChannel (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the channel.
- int TvDeviceSetVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the volume, update the TvDevice control service state table, and notify all subscribed control points of the updated state.
- int TvDeviceIncreaseVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the volume.
- int TvDeviceDecreaseVolume (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the volume.
- int TvDeviceSetColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the color, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- int TvDeviceIncreaseColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the color.
- int TvDeviceDecreaseColor (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the color.
- int TvDeviceSetTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the tint, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- int TvDeviceIncreaseTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase tint.
- int TvDeviceDecreaseTint (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease tint.
- int TvDeviceSetContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the contrast, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- int TvDeviceIncreaseContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase the contrast.
- int TvDeviceDecreaseContrast (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease the contrast.
- int TvDeviceSetBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Change the brightness, update the TvDevice picture service state table, and notify all subscribed control points of the updated state.
- int TvDeviceIncreaseBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Increase brightnesss.
- int TvDeviceDecreaseBrightness (IXML_Document *in, IXML_Document **out, const char **errorString)

  Decrease brightnesss.
- int TvDeviceStart (char *iface, unsigned short port, const char *desc_doc_name, const char *web_dir↩ _path, int ip_mode, print_string pfun, int combo)

  Initializes the UPnP Sdk, registers the device, and sends out advertisements.
- int TvDeviceStop (void)

  Stops the device. Uninitializes the sdk.
- void * TvDeviceCommandLoop (void *args)

  Function that receives commands from the user at the command prompt during the lifetime of the device, and calls the appropriate functions for those commands. Only one command, exit, is currently defined.
- int device_main (int argc, char *argv[])

  Main entry point for tv device application.

Device Sample API

- #define MAX_COLOR 10
- #define **MIN_COLOR** 1
- #define MAX_BRIGHTNESS 10
- #define **MIN_BRIGHTNESS** 1
- #define POWER_ON 1
- #define **POWER_OFF** 0
- #define MAX_TINT 10
- #define **MIN_TINT** 1
- #define MAX_VOLUME 10
- #define **MIN_VOLUME** 1
- #define MAX_CONTRAST 10
- #define **MIN_CONTRAST** 1
- #define MAX_CHANNEL 100
- #define **MIN_CHANNEL** 1
- #define TV_SERVICE_SERVCOUNT 2
- #define TV_SERVICE_CONTROL 0
- #define TV_SERVICE_PICTURE 1
- #define TV_CONTROL_VARCOUNT 3
- #define TV_CONTROL_POWER 0
- #define TV_CONTROL_CHANNEL 1
- #define TV_CONTROL_VOLUME 2
- #define TV_PICTURE_VARCOUNT 4
- #define TV_PICTURE_COLOR 0
- #define TV_PICTURE_TINT 1
- #define TV_PICTURE_CONTRAST 2
- #define TV_PICTURE_BRIGHTNESS 3
- #define TV_MAX_VAL_LEN 5
- #define TV_MAXACTIONS 12
- #define TV_MAXVARS TV_PICTURE_VARCOUNT
- #define **IP_MODE_IPV4** 1
- #define **IP_MODE_IPV6_LLA** 2
- #define **IP_MODE_IPV6_ULA_GUA** 3
- typedef int(∗ upnp_action) (IXML_Document ∗request, IXML_Document ∗∗out, const char ∗∗errorString)

  Prototype for all actions. For each action that a service implements, there is a corresponding function with this prototype.
- struct TvService tv_service_table [ ]
- UpnpDevice_Handle device_handle
- ithread_mutex_t TVDevMutex

9.72 tv_device.h

Go to the documentation of this file.

```
00001 #ifndef UPNP_TV_DEVICE_H
00002 #define UPNP_TV_DEVICE_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  *
00009  * Redistribution and use in source and binary forms, with or without
00010  * modification, are permitted provided that the following conditions are met:
00011  *
00012  * - Redistributions of source code must retain the above copyright notice,
00013  * this list of conditions and the following disclaimer.
00014  * - Redistributions in binary form must reproduce the above copyright notice,
```

```
00015  * this list of conditions and the following disclaimer in the documentation
00016  * and/or other materials provided with the distribution.
00017  * - Neither name of Intel Corporation nor the names of its contributors
00018  * may be used to endorse or promote products derived from this software
00019  * without specific prior written permission.
00020  *
00021  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00022  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00023  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00024  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00025  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00026  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00027  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00028  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00029  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00030  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00031  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00032  *
00033  ******************************************************************************/
00034
00047 #include <signal.h>
00048 #include <stdio.h>
00049
00050 #ifdef __cplusplus
00051 extern "C" {
00052 #endif
00053
00054 #include "sample_util.h"
00055
00056 #include "ithread.h"
00057 #include "upnp.h"
00058
00059 #include <stdlib.h>
00060 #include <string.h>
00061
00063 #define MAX_COLOR 10
00064 #define MIN_COLOR 1
00065
00067 #define MAX_BRIGHTNESS 10
00068 #define MIN_BRIGHTNESS 1
00069
00071 #define POWER_ON 1
00072 #define POWER_OFF 0
00073
00075 #define MAX_TINT 10
00076 #define MIN_TINT 1
00077
00079 #define MAX_VOLUME 10
00080 #define MIN_VOLUME 1
00081
00083 #define MAX_CONTRAST 10
00084 #define MIN_CONTRAST 1
00085
00087 #define MAX_CHANNEL 100
00088 #define MIN_CHANNEL 1
00089
00091 #define TV_SERVICE_SERVCOUNT 2
00092
00094 #define TV_SERVICE_CONTROL 0
00095
00097 #define TV_SERVICE_PICTURE 1
00098
00100 #define TV_CONTROL_VARCOUNT 3
00101
00103 #define TV_CONTROL_POWER 0
00104
00106 #define TV_CONTROL_CHANNEL 1
00107
00109 #define TV_CONTROL_VOLUME 2
00110
00112 #define TV_PICTURE_VARCOUNT 4
00113
00115 #define TV_PICTURE_COLOR 0
00116
00118 #define TV_PICTURE_TINT 1
00119
00121 #define TV_PICTURE_CONTRAST 2
00122
00124 #define TV_PICTURE_BRIGHTNESS 3
00125
00127 #define TV_MAX_VAL_LEN 5
00128
00130 #define TV_MAXACTIONS 12
00131
00133 #define TV_MAXVARS TV_PICTURE_VARCOUNT
00134
00135 #define IP_MODE_IPV4 1
```

```
00136 #define IP_MODE_IPV6_LLA 2
00137 #define IP_MODE_IPV6_ULA_GUA 3
00138
00149 typedef int (*upnp_action)(
00151         IXML_Document *request,
00153         IXML_Document **out,
00155         const char **errorString);
00156
00158 struct TvService
00159 {
00161         char UDN[NAME_SIZE];
00163         char ServiceId[NAME_SIZE];
00165         char ServiceType[NAME_SIZE];
00167         const char *VariableName[TV_MAXVARS];
00169         char *VariableStrVal[TV_MAXVARS];
00171         const char *ActionNames[TV_MAXACTIONS];
00173         upnp_action actions[TV_MAXACTIONS];
00175         int VariableCount;
00176 };
00177
00179 extern struct TvService tv_service_table[];
00180
00182 extern UpnpDevice_Handle device_handle;
00183
00188 extern ithread_mutex_t TVDevMutex;
00189
00196 int SetActionTable(
00198         int serviceType,
00200         struct TvService *out);
00201
00210 int TvDeviceStateTableInit(
00212         char *DescDocURL);
00213
00220 int TvDeviceHandleSubscriptionRequest(
00222         const UpnpSubscriptionRequest *sr_event);
00223
00230 int TvDeviceHandleGetVarRequest(
00232         UpnpStateVarRequest *cgv_event);
00233
00240 int TvDeviceHandleActionRequest(
00242         UpnpActionRequest *ca_event);
00243
00255 int TvDeviceCallbackEventHandler(
00257         Upnp_EventType,
00259         const void *Event,
00261         void *Cookie);
00262
00271 int TvDeviceSetServiceTableVar(
00274         unsigned int service,
00278         int variable,
00280         char *value);
00281
00282 /* Control Service Actions */
00283
00287 int TvDevicePowerOn(
00289         IXML_Document *in,
00291         IXML_Document **out,
00293         const char **errorString);
00294
00298 int TvDevicePowerOff(
00300         IXML_Document *in,
00302         IXML_Document **out,
00304         const char **errorString);
00305
00311 int TvDeviceSetChannel(
00313         IXML_Document *in,
00315         IXML_Document **out,
00317         const char **errorString);
00318
00322 int TvDeviceIncreaseChannel(
00324         IXML_Document *in,
00326         IXML_Document **out,
00328         const char **errorString);
00329
00333 int TvDeviceDecreaseChannel(
00335         IXML_Document *in,
00337         IXML_Document **out,
00339         const char **errorString);
00340
00346 int TvDeviceSetVolume(
00348         IXML_Document *in,
00350         IXML_Document **out,
00352         const char **errorString);
00353
00357 int TvDeviceIncreaseVolume(
00359         IXML_Document *in,
00361         IXML_Document **out,
```

```
00363         const char **errorString);
00364
00368 int TvDeviceDecreaseVolume(
00370         IXML_Document *in,
00372         IXML_Document **out,
00374         const char **errorString);
00375
00376 /*Picture Service Actions */
00377
00383 int TvDeviceSetColor(
00385         IXML_Document *in,
00387         IXML_Document **out,
00389         const char **errorString);
00390
00394 int TvDeviceIncreaseColor(
00396         IXML_Document *in,
00398         IXML_Document **out,
00400         const char **errorString);
00401
00405 int TvDeviceDecreaseColor(
00407         IXML_Document *in,
00409         IXML_Document **out,
00411         const char **errorString);
00412
00418 int TvDeviceSetTint(
00420         IXML_Document *in,
00422         IXML_Document **out,
00424         const char **errorString);
00425
00429 int TvDeviceIncreaseTint(
00431         IXML_Document *in,
00433         IXML_Document **out,
00435         const char **errorString);
00436
00440 int TvDeviceDecreaseTint(
00442         IXML_Document *in,
00444         IXML_Document **out,
00446         const char **errorString);
00447
00453 int TvDeviceSetContrast(
00455         IXML_Document *in,
00457         IXML_Document **out,
00459         const char **errorString);
00460
00464 int TvDeviceIncreaseContrast(
00466         IXML_Document *in,
00468         IXML_Document **out,
00470         const char **errorString);
00471
00475 int TvDeviceDecreaseContrast(
00477         IXML_Document *in,
00479         IXML_Document **out,
00481         const char **errorString);
00482
00488 int TvDeviceSetBrightness(
00490         IXML_Document *in,
00492         IXML_Document **out,
00494         const char **errorString);
00495
00499 int TvDeviceIncreaseBrightness(
00501         IXML_Document *in,
00503         IXML_Document **out,
00505         const char **errorString);
00506
00510 int TvDeviceDecreaseBrightness(
00512         IXML_Document *in,
00514         IXML_Document **out,
00516         const char **errorString);
00517
00522 int TvDeviceStart(
00525         char *iface,
00528         unsigned short port,
00531         const char *desc_doc_name,
00534         const char *web_dir_path,
00537         int ip_mode,
00539         print_string pfun,
00541         int combo);
00542
00546 int TvDeviceStop(void);
00547
00554 void *TvDeviceCommandLoop(void *args);
00555
00570 int device_main(int argc, char *argv[]);
00571
00572 #ifdef __cplusplus
00573 }
00574 #endif
```

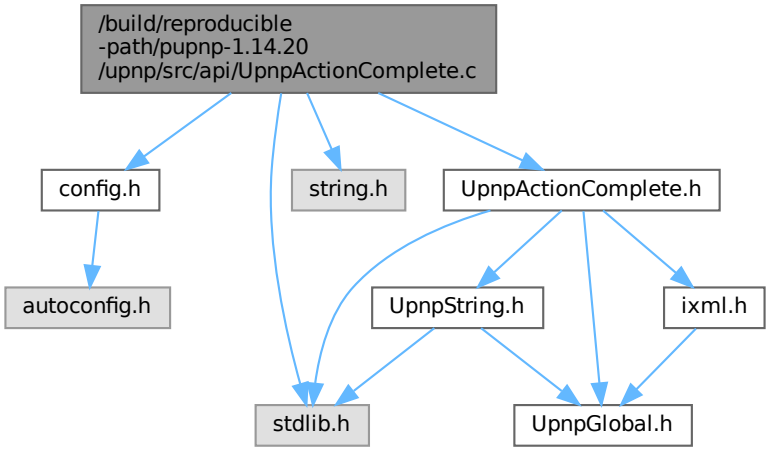
```
00575
00580 #endif /* UPNP_TV_DEVICE_H */
```

9.73 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionComplete.c File Reference

Source file for UpnpActionComplete methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpActionComplete.h"
```

Include dependency graph for UpnpActionComplete.c:



Data Structures

- struct s_UpnpActionComplete

Functions

- UpnpActionComplete * UpnpActionComplete_new ()
- void UpnpActionComplete_delete (UpnpActionComplete *q)
- int UpnpActionComplete_assign (UpnpActionComplete *p, const UpnpActionComplete *q)
- UpnpActionComplete * UpnpActionComplete_dup (const UpnpActionComplete *q)
- int UpnpActionComplete_get_ErrCode (const UpnpActionComplete *p)
- int UpnpActionComplete_set_ErrCode (UpnpActionComplete *p, int n)
- const UpnpString * UpnpActionComplete_get_CtrlUrl (const UpnpActionComplete *p)
- int UpnpActionComplete_set_CtrlUrl (UpnpActionComplete *p, const UpnpString *s)
- size_t UpnpActionComplete_get_CtrlUrl_Length (const UpnpActionComplete *p)
- const char * UpnpActionComplete_get_CtrlUrl_cstr (const UpnpActionComplete *p)
- int UpnpActionComplete_strcpy_CtrlUrl (UpnpActionComplete *p, const char *s)

- int UpnpActionComplete_strncpy_CtrlUrl (UpnpActionComplete *p, const char *s, size_t n)
- void UpnpActionComplete_clear_CtrlUrl (UpnpActionComplete *p)
- IXML_Document * UpnpActionComplete_get_ActionRequest (const UpnpActionComplete *p)
- int UpnpActionComplete_set_ActionRequest (UpnpActionComplete *p, IXML_Document *n)
- IXML_Document * UpnpActionComplete_get_ActionResult (const UpnpActionComplete *p)
- int UpnpActionComplete_set_ActionResult (UpnpActionComplete *p, IXML_Document *n)

9.73.1 Detailed Description

Source file for UpnpActionComplete methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.73.2 Function Documentation

9.73.2.1 UpnpActionComplete_assign()

```
int UpnpActionComplete_assign (
            UpnpActionComplete * p,
            const UpnpActionComplete * q )
```

Assignment operator

References UpnpActionComplete_get_ActionRequest(), UpnpActionComplete_get_ActionResult(), UpnpActionComplete_get_CtrlUrl(), UpnpActionComplete_get_ErrCode(), UpnpActionComplete_set_ActionRequest(), UpnpActionComplete_set_ActionResult(), UpnpActionComplete_set_CtrlUrl(), and UpnpActionComplete_set_ErrCode().

Referenced by UpnpActionComplete_dup().

9.73.2.2 UpnpActionComplete_clear_CtrlUrl()

```
void UpnpActionComplete_clear_CtrlUrl (
            UpnpActionComplete * p )
```

UpnpActionComplete_clear_CtrlUrl

References UpnpString_clear().

9.73.2.3 UpnpActionComplete_delete()

```
void UpnpActionComplete_delete (
            UpnpActionComplete * p )
```

Destructor

References UpnpString_delete().

Referenced by TestClass_delete(), TestClass_set_TheActionComplete(), and UpnpThreadDistribution().

9.73.2.4 UpnpActionComplete_dup()

```
UpnpActionComplete * UpnpActionComplete_dup (
            const UpnpActionComplete * p )
```

Copy Constructor

References UpnpActionComplete_assign(), and UpnpActionComplete_new().

Referenced by TestClass_set_TheActionComplete().

9.73.2.5 UpnpActionComplete_get_ActionRequest()

```
IXML_Document * UpnpActionComplete_get_ActionRequest (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_ActionRequest

Referenced by SampleUtil_PrintEvent(), and UpnpActionComplete_assign().

9.73.2.6 UpnpActionComplete_get_ActionResult()

```
IXML_Document * UpnpActionComplete_get_ActionResult (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_ActionResult

Referenced by SampleUtil_PrintEvent(), and UpnpActionComplete_assign().

9.73.2.7 UpnpActionComplete_get_CtrlUrl()

```
const UpnpString * UpnpActionComplete_get_CtrlUrl (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_CtrlUrl

Referenced by SampleUtil_PrintEvent(), UpnpActionComplete_assign(), UpnpActionComplete_get_CtrlUrl_cstr(), and UpnpActionComplete_get_CtrlUrl_Length().

9.73.2.8 UpnpActionComplete_get_CtrlUrl_cstr()

```
const char * UpnpActionComplete_get_CtrlUrl_cstr (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_CtrlUrl_cstr

References UpnpActionComplete_get_CtrlUrl(), and UpnpString_get_String().

9.73.2.9 UpnpActionComplete_get_CtrlUrl_Length()

```
size_t UpnpActionComplete_get_CtrlUrl_Length (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_CtrlUrl_Length

References UpnpActionComplete_get_CtrlUrl(), and UpnpString_get_Length().

9.73.2.10 UpnpActionComplete_get_ErrCode()

```
int UpnpActionComplete_get_ErrCode (
            const UpnpActionComplete * p )
```

UpnpActionComplete_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpActionComplete_assign().

9.73.2.11 UpnpActionComplete_new()

```
UpnpActionComplete * UpnpActionComplete_new ( )
```

Constructor

References UpnpString_new().

Referenced by TestClass_new(), UpnpActionComplete_dup(), and UpnpThreadDistribution().

9.73.2.12 UpnpActionComplete_set_ActionRequest()

```
int UpnpActionComplete_set_ActionRequest (
            UpnpActionComplete * p,
            IXML_Document * n )
```

UpnpActionComplete_set_ActionRequest

Referenced by UpnpActionComplete_assign(), and UpnpThreadDistribution().

9.73.2.13 UpnpActionComplete_set_ActionResult()

```
int UpnpActionComplete_set_ActionResult (
            UpnpActionComplete * p,
            IXML_Document * n )
```

UpnpActionComplete_set_ActionResult

Referenced by UpnpActionComplete_assign(), and UpnpThreadDistribution().

9.73.2.14 UpnpActionComplete_set_CtrlUrl()

```
int UpnpActionComplete_set_CtrlUrl (
            UpnpActionComplete * p,
            const UpnpString * s )
```

UpnpActionComplete_set_CtrlUrl

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionComplete_assign().

9.73.2.15 UpnpActionComplete_set_ErrCode()

```
int UpnpActionComplete_set_ErrCode (
            UpnpActionComplete * p,
            int n )
```

UpnpActionComplete_set_ErrCode

Referenced by UpnpActionComplete_assign(), and UpnpThreadDistribution().

9.73.2.16 UpnpActionComplete_strcpy_CtrlUrl()

```
int UpnpActionComplete_strcpy_CtrlUrl (
            UpnpActionComplete * p,
            const char * s )
```

UpnpActionComplete_strcpy_CtrlUrl

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.73.2.17 UpnpActionComplete_strncpy_CtrlUrl()

```
int UpnpActionComplete_strncpy_CtrlUrl (
            UpnpActionComplete * p,
            const char * s,
            size_t n )
```

UpnpActionComplete_strncpy_CtrlUrl

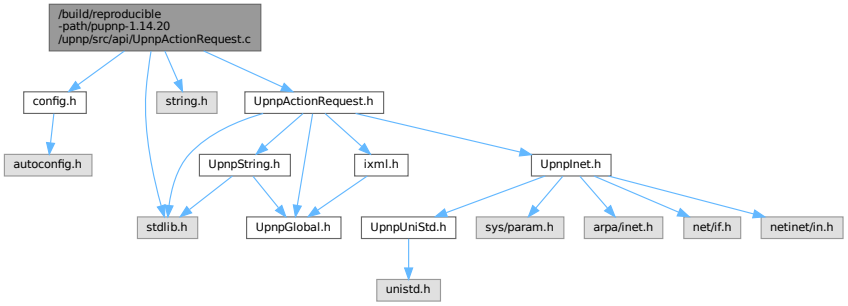
References UpnpString_set_StringN().

9.74 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionRequest.c File Reference

Source file for UpnpActionRequest methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpActionRequest.h"
```

Include dependency graph for UpnpActionRequest.c:



Data Structures

- struct s_UpnpActionRequest

Functions

- UpnpActionRequest * UpnpActionRequest_new ()
- void UpnpActionRequest_delete (UpnpActionRequest *q)
- int UpnpActionRequest_assign (UpnpActionRequest *p, const UpnpActionRequest *q)
- UpnpActionRequest * UpnpActionRequest_dup (const UpnpActionRequest *q)
- int UpnpActionRequest_get_ErrCode (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ErrCode (UpnpActionRequest *p, int n)
- int UpnpActionRequest_get_Socket (const UpnpActionRequest *p)
- int UpnpActionRequest_set_Socket (UpnpActionRequest *p, int n)
- const UpnpString * UpnpActionRequest_get_ErrStr (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ErrStr (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_ErrStr_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_ErrStr_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_ErrStr (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_ErrStr (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_ErrStr (UpnpActionRequest *p)
- const UpnpString * UpnpActionRequest_get_ActionName (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ActionName (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_ActionName_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_ActionName_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_ActionName (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_ActionName (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_ActionName (UpnpActionRequest *p)

- const UpnpString * UpnpActionRequest_get_DevUDN (const UpnpActionRequest *p)
- int UpnpActionRequest_set_DevUDN (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_DevUDN_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_DevUDN_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_DevUDN (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_DevUDN (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_DevUDN (UpnpActionRequest *p)
- const UpnpString * UpnpActionRequest_get_ServiceID (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ServiceID (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_ServiceID_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_ServiceID_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_ServiceID (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_ServiceID (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_ServiceID (UpnpActionRequest *p)
- IXML_Document * UpnpActionRequest_get_ActionRequest (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ActionRequest (UpnpActionRequest *p, IXML_Document *n)
- IXML_Document * UpnpActionRequest_get_ActionResult (const UpnpActionRequest *p)
- int UpnpActionRequest_set_ActionResult (UpnpActionRequest *p, IXML_Document *n)
- IXML_Document * UpnpActionRequest_get_SoapHeader (const UpnpActionRequest *p)
- int UpnpActionRequest_set_SoapHeader (UpnpActionRequest *p, IXML_Document *n)
- const struct sockaddr_storage * UpnpActionRequest_get_CtrlPtIPAddr (const UpnpActionRequest *p)
- int UpnpActionRequest_set_CtrlPtIPAddr (UpnpActionRequest *p, const struct sockaddr_storage *buf)
- void UpnpActionRequest_clear_CtrlPtIPAddr (UpnpActionRequest *p)
- const UpnpString * UpnpActionRequest_get_Os (const UpnpActionRequest *p)
- int UpnpActionRequest_set_Os (UpnpActionRequest *p, const UpnpString *s)
- size_t UpnpActionRequest_get_Os_Length (const UpnpActionRequest *p)
- const char * UpnpActionRequest_get_Os_cstr (const UpnpActionRequest *p)
- int UpnpActionRequest_strcpy_Os (UpnpActionRequest *p, const char *s)
- int UpnpActionRequest_strncpy_Os (UpnpActionRequest *p, const char *s, size_t n)
- void UpnpActionRequest_clear_Os (UpnpActionRequest *p)

9.74.1 Detailed Description

Source file for UpnpActionRequest methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.74.2 Function Documentation

9.74.2.1 UpnpActionRequest_assign()

```
int UpnpActionRequest_assign (
            UpnpActionRequest * p,
            const UpnpActionRequest * q )
```

Assignment operator

References UpnpActionRequest_get_ActionName(), UpnpActionRequest_get_ActionRequest(), UpnpActionRequest_get_ActionResu
UpnpActionRequest_get_CtrlPtIPAddr(), UpnpActionRequest_get_DevUDN(), UpnpActionRequest_get_ErrCode(),
UpnpActionRequest_get_ErrStr(), UpnpActionRequest_get_Os(), UpnpActionRequest_get_ServiceID(), UpnpActionRequest_get_So
UpnpActionRequest_get_Socket(), UpnpActionRequest_set_ActionName(), UpnpActionRequest_set_ActionRequest(),
UpnpActionRequest_set_ActionResult(), UpnpActionRequest_set_CtrlPtIPAddr(), UpnpActionRequest_set_DevUDN(),
UpnpActionRequest_set_ErrCode(), UpnpActionRequest_set_ErrStr(), UpnpActionRequest_set_Os(), UpnpActionRequest_set_Serv
UpnpActionRequest_set_SoapHeader(), and UpnpActionRequest_set_Socket().

Referenced by UpnpActionRequest_dup().

9.74.2.2 UpnpActionRequest_clear_ActionName()

```
void UpnpActionRequest_clear_ActionName (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_ActionName

References UpnpString_clear().

9.74.2.3 UpnpActionRequest_clear_CtrlPtIPAddr()

```
void UpnpActionRequest_clear_CtrlPtIPAddr (
            UpnpActionRequest * p )
```

UpnpActionRequest_get_CtrlPtIPAddr

9.74.2.4 UpnpActionRequest_clear_DevUDN()

```
void UpnpActionRequest_clear_DevUDN (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_DevUDN

References UpnpString_clear().

9.74.2.5 UpnpActionRequest_clear_ErrStr()

```
void UpnpActionRequest_clear_ErrStr (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_ErrStr

References UpnpString_clear().

9.74.2.6 UpnpActionRequest_clear_Os()

```
void UpnpActionRequest_clear_Os (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_Os

References UpnpString_clear().

9.74.2.7 UpnpActionRequest_clear_ServiceID()

```
void UpnpActionRequest_clear_ServiceID (
            UpnpActionRequest * p )
```

UpnpActionRequest_clear_ServiceID

References UpnpString_clear().

9.74.2.8 UpnpActionRequest_delete()

```
void UpnpActionRequest_delete (
            UpnpActionRequest * p )
```

Destructor

References UpnpString_delete().

Referenced by handle_invoke_action().

9.74.2.9 UpnpActionRequest_dup()

```
UpnpActionRequest * UpnpActionRequest_dup (
            const UpnpActionRequest * p )
```

Copy Constructor

References UpnpActionRequest_assign(), and UpnpActionRequest_new().

9.74.2.10 UpnpActionRequest_get_ActionName()

```
const UpnpString * UpnpActionRequest_get_ActionName (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionName

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), UpnpActionRequest_assign(), UpnpActionRequest_get_ActionName_cstr(), and UpnpActionRequest_get_ActionName_Length().

9.74.2.11 UpnpActionRequest_get_ActionName_cstr()

```
const char * UpnpActionRequest_get_ActionName_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionName_cstr

References UpnpActionRequest_get_ActionName(), and UpnpString_get_String().

9.74.2.12 UpnpActionRequest_get_ActionName_Length()

```
size_t UpnpActionRequest_get_ActionName_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionName_Length

References UpnpActionRequest_get_ActionName(), and UpnpString_get_Length().

9.74.2.13 UpnpActionRequest_get_ActionRequest()

```
IXML_Document * UpnpActionRequest_get_ActionRequest (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionRequest

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.74.2.14 UpnpActionRequest_get_ActionResult()

```
IXML_Document * UpnpActionRequest_get_ActionResult (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ActionResult

Referenced by handle_invoke_action(), SampleUtil_PrintEvent(), and UpnpActionRequest_assign().

9.74.2.15 UpnpActionRequest_get_CtrlPtIPAddr()

```
const struct sockaddr_storage * UpnpActionRequest_get_CtrlPtIPAddr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_CtrlPtIPAddr

Referenced by UpnpActionRequest_assign().

9.74.2.16 UpnpActionRequest_get_DevUDN()

```
const UpnpString * UpnpActionRequest_get_DevUDN (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_DevUDN

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), UpnpActionRequest_assign(), UpnpActionRequest_get_DevUDN_cstr(), and UpnpActionRequest_get_DevUDN_Length().

9.74.2.17 UpnpActionRequest_get_DevUDN_cstr()

```
const char * UpnpActionRequest_get_DevUDN_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_DevUDN_cstr

References UpnpActionRequest_get_DevUDN(), and UpnpString_get_String().

9.74.2.18 UpnpActionRequest_get_DevUDN_Length()

```
size_t UpnpActionRequest_get_DevUDN_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_DevUDN_Length

References UpnpActionRequest_get_DevUDN(), and UpnpString_get_Length().

9.74.2.19 UpnpActionRequest_get_ErrCode()

```
int UpnpActionRequest_get_ErrCode (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrCode

Referenced by handle_invoke_action(), SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.74.2.20 UpnpActionRequest_get_ErrStr()

```
const UpnpString * UpnpActionRequest_get_ErrStr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrStr

Referenced by SampleUtil_PrintEvent(), UpnpActionRequest_assign(), UpnpActionRequest_get_ErrStr_cstr(), and UpnpActionRequest_get_ErrStr_Length().

9.74.2.21 UpnpActionRequest_get_ErrStr_cstr()

```
const char * UpnpActionRequest_get_ErrStr_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrStr_cstr

References UpnpActionRequest_get_ErrStr(), and UpnpString_get_String().

Referenced by handle_invoke_action().

9.74.2.22 UpnpActionRequest_get_ErrStr_Length()

```
size_t UpnpActionRequest_get_ErrStr_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ErrStr_Length

References UpnpActionRequest_get_ErrStr(), and UpnpString_get_Length().

9.74.2.23 UpnpActionRequest_get_Os()

```
const UpnpString * UpnpActionRequest_get_Os (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Os

Referenced by UpnpActionRequest_assign(), UpnpActionRequest_get_Os_cstr(), and UpnpActionRequest_get_Os_Length().

9.74.2.24 UpnpActionRequest_get_Os_cstr()

```
const char * UpnpActionRequest_get_Os_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Os_cstr

References UpnpActionRequest_get_Os(), and UpnpString_get_String().

9.74.2.25 UpnpActionRequest_get_Os_Length()

```
size_t UpnpActionRequest_get_Os_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Os_Length

References UpnpActionRequest_get_Os(), and UpnpString_get_Length().

9.74.2.26 UpnpActionRequest_get_ServiceID()

```
const UpnpString * UpnpActionRequest_get_ServiceID (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ServiceID

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleActionRequest(), UpnpActionRequest_assign(), UpnpActionRequest_get_ServiceID_cstr(), and UpnpActionRequest_get_ServiceID_Length().

9.74.2.27 UpnpActionRequest_get_ServiceID_cstr()

```
const char * UpnpActionRequest_get_ServiceID_cstr (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ServiceID_cstr

References UpnpActionRequest_get_ServiceID(), and UpnpString_get_String().

9.74.2.28 UpnpActionRequest_get_ServiceID_Length()

```
size_t UpnpActionRequest_get_ServiceID_Length (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_ServiceID_Length

References UpnpActionRequest_get_ServiceID(), and UpnpString_get_Length().

9.74.2.29 UpnpActionRequest_get_SoapHeader()

```
IXML_Document * UpnpActionRequest_get_SoapHeader (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_SoapHeader

Referenced by UpnpActionRequest_assign().

9.74.2.30 UpnpActionRequest_get_Socket()

```
int UpnpActionRequest_get_Socket (
            const UpnpActionRequest * p )
```

UpnpActionRequest_get_Socket

Referenced by UpnpActionRequest_assign().

9.74.2.31 UpnpActionRequest_new()

```
UpnpActionRequest * UpnpActionRequest_new ( )
```

Constructor

References UpnpString_new().

Referenced by handle_invoke_action(), and UpnpActionRequest_dup().

9.74.2.32 UpnpActionRequest_set_ActionName()

```
int UpnpActionRequest_set_ActionName (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_ActionName

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.74.2.33 UpnpActionRequest_set_ActionRequest()

```
int UpnpActionRequest_set_ActionRequest (
            UpnpActionRequest * p,
            IXML_Document * n )
```

UpnpActionRequest_set_ActionRequest

Referenced by handle_invoke_action(), and UpnpActionRequest_assign().

9.74.2.34 UpnpActionRequest_set_ActionResult()

```
int UpnpActionRequest_set_ActionResult (
            UpnpActionRequest * p,
            IXML_Document * n )
```

UpnpActionRequest_set_ActionResult

Referenced by handle_invoke_action(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.74.2.35 UpnpActionRequest_set_CtrlPtIPAddr()

```
int UpnpActionRequest_set_CtrlPtIPAddr (
            UpnpActionRequest * p,
            const struct sockaddr_storage * buf )
```

UpnpActionRequest_get_CtrlPtIPAddr

Referenced by handle_invoke_action(), and UpnpActionRequest_assign().

9.74.2.36 UpnpActionRequest_set_DevUDN()

```
int UpnpActionRequest_set_DevUDN (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_DevUDN

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.74.2.37 UpnpActionRequest_set_ErrCode()

```
int UpnpActionRequest_set_ErrCode (
            UpnpActionRequest * p,
            int n )
```

UpnpActionRequest_set_ErrCode

Referenced by handle_invoke_action(), TvDeviceHandleActionRequest(), and UpnpActionRequest_assign().

9.74.2.38 UpnpActionRequest_set_ErrStr()

```
int UpnpActionRequest_set_ErrStr (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_ErrStr

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.74.2.39 UpnpActionRequest_set_Os()

```
int UpnpActionRequest_set_Os (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_Os

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.74.2.40 UpnpActionRequest_set_ServiceID()

```
int UpnpActionRequest_set_ServiceID (
            UpnpActionRequest * p,
            const UpnpString * s )
```

UpnpActionRequest_set_ServiceID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpActionRequest_assign().

9.74.2.41 UpnpActionRequest_set_SoapHeader()

```
int UpnpActionRequest_set_SoapHeader (
            UpnpActionRequest * p,
            IXML_Document * n )
```

UpnpActionRequest_set_SoapHeader

Referenced by UpnpActionRequest_assign().

9.74.2.42 UpnpActionRequest_set_Socket()

```
int UpnpActionRequest_set_Socket (
            UpnpActionRequest * p,
            int n )
```

UpnpActionRequest_set_Socket

Referenced by UpnpActionRequest_assign().

9.74.2.43 UpnpActionRequest_strcpy_ActionName()

```
int UpnpActionRequest_strcpy_ActionName (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_ActionName

References UpnpString_set_String().

Referenced by handle_invoke_action().

9.74.2.44 UpnpActionRequest_strcpy_DevUDN()

```
int UpnpActionRequest_strcpy_DevUDN (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_DevUDN

References UpnpString_set_String().

Referenced by handle_invoke_action().

9.74.2.45 UpnpActionRequest_strcpy_ErrStr()

```
int UpnpActionRequest_strcpy_ErrStr (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_ErrStr

References UpnpString_set_String().

Referenced by TvDeviceHandleActionRequest().

9.74.2.46 UpnpActionRequest_strcpy_Os()

```
int UpnpActionRequest_strcpy_Os (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_Os

References UpnpString_set_String().

9.74.2.47 UpnpActionRequest_strcpy_ServiceID()

```
int UpnpActionRequest_strcpy_ServiceID (
            UpnpActionRequest * p,
            const char * s )
```

UpnpActionRequest_strcpy_ServiceID

References UpnpString_set_String().

Referenced by handle_invoke_action().

9.74.2.48 UpnpActionRequest_strncpy_ActionName()

```
int UpnpActionRequest_strncpy_ActionName (
            UpnpActionRequest * p,
            const char * s,
            size_t n )
```

UpnpActionRequest_strncpy_ActionName

References UpnpString_set_StringN().

9.74.2.49 UpnpActionRequest_strncpy_DevUDN()

```
int UpnpActionRequest_strncpy_DevUDN (
            UpnpActionRequest * p,
            const char * s,
            size_t n )
```

UpnpActionRequest_strncpy_DevUDN

References UpnpString_set_StringN().

9.74.2.50 UpnpActionRequest_strncpy_ErrStr()

```
int UpnpActionRequest_strncpy_ErrStr (
            UpnpActionRequest * p,
            const char * s,
            size_t n )
```

UpnpActionRequest_strncpy_ErrStr

References UpnpString_set_StringN().

9.74.2.51 UpnpActionRequest_strncpy_Os()

```
int UpnpActionRequest_strncpy_Os (
            UpnpActionRequest * p,
            const char * s,
            size_t n )
```

UpnpActionRequest_strncpy_Os

References UpnpString_set_StringN().

Referenced by handle_invoke_action().

9.74.2.52 UpnpActionRequest_strncpy_ServiceID()

```
int UpnpActionRequest_strncpy_ServiceID (
            UpnpActionRequest * p,
            const char * s,
            size_t n )
```

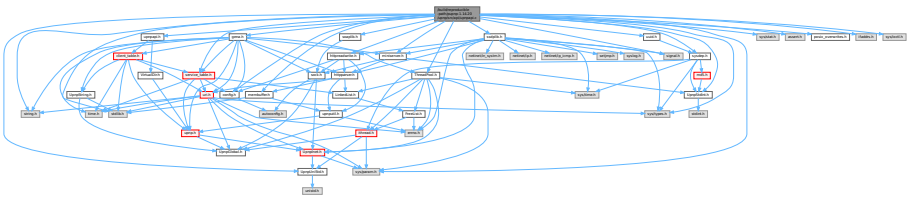
UpnpActionRequest_strncpy_ServiceID

References UpnpString_set_StringN().

9.75 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpapi.c File Reference

```
#include "config.h"
#include "upnpapi.h"
#include "ThreadPool.h"
#include "UpnpStdInt.h"
#include "UpnpUniStd.h"
#include "httpreadwrite.h"
#include "membuffer.h"
#include "soaplib.h"
#include "ssdplib.h"
#include "sysdep.h"
#include "uuid.h"
#include "gena.h"
#include "miniserver.h"
#include "service_table.h"
#include <sys/stat.h>
#include <assert.h>
#include <signal.h>
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
#include <ifaddrs.h>
#include <sys/ioctl.h>
#include <sys/param.h>
#include <sys/types.h>
```

Include dependency graph for upnpapi.c:



Data Structures

- struct job_arg

Macros

- #define **ifr_netmask** ifr_addr
- #define IN6_IS_ADDR_GLOBAL(a)
- #define IN6_IS_ADDR_ULA(a)

Functions

- static void free_advertise_arg (job_arg *arg)

  Free memory associated with advertise job's argument.
- static void free_action_arg (job_arg *arg)

  Free memory associated with an action job's argument.
- static int WinsockInit (void)

  (Windows Only) Initializes the Windows Winsock library.
- static int UpnpInitMutexes (void)

  Initializes the global mutexes used by the UPnP SDK.
- static int UpnpInitThreadPools (void)

  Initializes the global threadm pools used by the UPnP SDK.
- static int UpnpInitPreamble (void)

  Performs the initial steps in initializing the UPnP SDK.
- static int UpnpInitStartServers (unsigned short DestPort)

  Finishes initializing the UPnP SDK.
- void PrintThreadPoolStats (ThreadPool *tp, const char *DbgFileName, int DbgLineNo, const char *msg)

  Prints thread pool statistics.
- static int GetFreeHandle ()

  Get a free handle.
- static int FreeHandle (int Upnp_Handle)

  Free handle.
- static int GetDescDocumentAndURL (Upnp_DescType descriptionType, char *description, int config_base↩ URL, int AddressFamily, IXML_Document **xmlDoc, char descURL[LINE_SIZE])

  Fills the sockadr_in with miniserver information.
- static unsigned UpnpComputeIpv6PrefixLength (struct sockaddr_in6 *Netmask)

  Computes prefix length from IPv6 netmask.
- int UpnpGetIfInfo (const char *IfName)

  Retrieve interface information and keep it in global variables. If NULL, we'll find the first suitable interface for operation.
- void UpnpThreadDistribution (struct UpnpNonblockParam *Param)

  Schedule async functions in threadpool.
- Upnp_FunPtr GetCallBackFn (UpnpClient_Handle Hnd)

  Get callback function ptr from a handle.
- Upnp_Handle_Type GetClientHandleInfo (UpnpClient_Handle *client_handle_out, struct Handle_Info **HndInfo)

  Get client handle info.
- Upnp_Handle_Type GetDeviceHandleInfo (UpnpDevice_Handle start, int AddressFamily, UpnpDevice_Handle *device_handle_out, struct Handle_Info **HndInfo)

  Retrieves the device handle and information of the first device of the address family specified. The search begins at the 'start' index, which should be 0 for the first call, then the last successful value returned. This allows listing all entries for the address family.
- Upnp_Handle_Type GetDeviceHandleInfoForPath (const char *path, int AddressFamily, UpnpDevice_Handle *device_handle_out, struct Handle_Info **HndInfo, service_info **serv_info)

  Retrieves the device handle and information of the first device of the address family specified, with a service having a controlURL or eventSubURL matching the path.
- Upnp_Handle_Type GetHandleInfo (UpnpClient_Handle Hnd, struct Handle_Info **HndInfo)

Get handle information.

- int PrintHandleInfo (UpnpClient_Handle Hnd)

  Print handle info.

- void AutoAdvertise (void *input)

  This function is a timer thread scheduled by UpnpSendAdvertisement to the send advetisement again.

- int **UpnpSetEventQueueLimits** (int maxLen, int maxAge)

Initialization and Registration

- int UpnpInit2 (const char *lfName, unsigned short DestPort)

  Initializes the Linux SDK for UPnP Devices (IPv4 or IPv6).

- int UpnpFinish (void)

  Initializes the OpenSSL library, and the OpenSSL context for use with pupnp.

- unsigned short UpnpGetServerPort (void)

  Returns the internal server IPv4 UPnP listening port.

- unsigned short UpnpGetServerPort6 (void)

  Returns the internal server IPv6 link-local (LLA) UPnP listening port.

- unsigned short UpnpGetServerUlaGuaPort6 (void)

  Returns the internal server IPv6 ULA or GUA UPnP listening port.

- char * UpnpGetServerIpAddress (void)

  Returns the local IPv4 listening ip address.

- char * UpnpGetServerIp6Address (void)

  Returns the IPv6 link-local listening ip address.

- char * UpnpGetServerUlaGuaIp6Address (void)

  Returns the IPv6 unique-local or globally-unique listening ip address.

- int UpnpRegisterRootDevice (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd)

  Registers a device application with the UPnP Library.

- int UpnpRegisterRootDevice2 (Upnp_DescType descriptionType, const char *description, size_t buffer↩ Len, int config_baseURL, Upnp_FunPtr Fun, const void *Cookie, UpnpDevice_Handle *Hnd)

  *Registers a device application with the UPnP Library. Similar to **UpnpRegisterRootDevice**, except that it also allows the description document to be specified as a file or a memory buffer.*

- int UpnpRegisterRootDevice3 (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd, int AddressFamily)

  Registers a device application for a specific address family with the UPnP library.

- int UpnpRegisterRootDevice4 (const char *DescUrl, Upnp_FunPtr Callback, const void *Cookie, UpnpDevice_Handle *Hnd, int AddressFamily, const char *LowerDescUrl)

  Registers a device application for a specific address family with the UPnP library. This function can also be used to specify a dedicated description URL to be returned for legacy CPs.

- int UpnpUnRegisterRootDevice (UpnpDevice_Handle Hnd)

  *Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **Upnp**↩ **RegisterRootDevice3** or **UpnpRegisterRootDevice4**.*

- int UpnpUnRegisterRootDeviceLowPower (UpnpDevice_Handle Hnd, int PowerState, int SleepPeriod, int RegistrationState)

  *Unregisters a root device registered with **UpnpRegisterRootDevice**, **UpnpRegisterRootDevice2**, **Upnp**↩ **RegisterRootDevice3** or **UpnpRegisterRootDevice4**.*

- int UpnpRegisterClient (Upnp_FunPtr Callback, const void *Cookie, UpnpClient_Handle *Hnd)

  Registers a control point application with the UPnP Library.

- int UpnpUnRegisterClient (UpnpClient_Handle Hnd)

  Unregisters a control point application, unsubscribing all active subscriptions.

- int UpnpSetContentLength (UpnpClient_Handle Hnd, size_t contentLength)
- int UpnpSetMaxContentLength (size_t contentLength)

  Sets the maximum content-length that the SDK will process on an incoming SOAP requests or responses.

Discovery

- int UpnpSendAdvertisement (UpnpDevice_Handle Hnd, int Exp)

Sends out the discovery announcements for all devices and services for a device.

- int UpnpSendAdvertisementLowPower (UpnpDevice_Handle Hnd, int Exp, int PowerState, int SleepPeriod, int RegistrationState)

  Sends out the discovery announcements for all devices and services for a device.
- int UpnpSearchAsync (UpnpClient_Handle Hnd, int Mx, const char *TTarget_constarget_const, const void *Cookie_const)

  Searches for devices matching the given search target.

Eventing

- int UpnpSetMaxSubscriptions (UpnpDevice_Handle Hnd, int MaxSubscriptions)

  Sets the maximum number of subscriptions accepted per service.
- int UpnpSetMaxSubscriptionTimeOut (UpnpDevice_Handle Hnd, int MaxSubscriptionTimeOut)

  Sets the maximum time-out accepted for a subscription request or renewal.
- int UpnpSubscribeAsync (UpnpClient_Handle Hnd, const char *PublisherUrl, int TimeOut, Upnp_FunPtr Fun, const void *Cookie)

  *Performs the same operation as **UpnpSubscribe**, but returns immediately and calls the registered callback function when the operation is complete.*
- int UpnpSubscribe (UpnpClient_Handle Hnd, const char *PublisherUrl, int *TimeOut, Upnp_SID SubsId)

  Registers a control point to receive event notifications from another device.
- int UpnpUnSubscribe (UpnpClient_Handle Hnd, const Upnp_SID SubsId)

  *Removes the subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**.*
- int UpnpUnSubscribeAsync (UpnpClient_Handle Hnd, Upnp_SID SubsId, Upnp_FunPtr Fun, const void *Cookie)

  *Removes a subscription of a control point from a service previously subscribed to using **UpnpSubscribe** or **UpnpSubscribeAsync**, generating a callback when the operation is complete.*
- int UpnpRenewSubscription (UpnpClient_Handle Hnd, int *TimeOut, const Upnp_SID SubsId)

  Renews a subscription that is about to expire.
- int UpnpRenewSubscriptionAsync (UpnpClient_Handle Hnd, int TimeOut, Upnp_SID SubsId, Upnp_FunPtr Fun, const void *Cookie)

  Renews a subscription that is about to expire, generating a callback when the operation is complete.
- int UpnpNotify (UpnpDevice_Handle, const char *DevID, const char *ServID, const char **VarName, const char **NewVal, int cVariables)

  Sends out an event change notification to all control points subscribed to a particular service.
- int UpnpNotifyExt (UpnpDevice_Handle, const char *DevID, const char *ServID, IXML_Document *PropSet)

  *Similar to **UpnpNotify** except that it takes a DOM document for the event rather than an array of strings.*
- int UpnpAcceptSubscription (UpnpDevice_Handle Hnd, const char *DevID, const char *ServID, const char **VarName, const char **NewVal, int cVariables, const Upnp_SID SubsId)

  Accepts a subscription request and sends out the current state of the eventable variables for a service.
- int UpnpAcceptSubscriptionExt (UpnpDevice_Handle Hnd, const char *DevID, const char *ServID, IXML_Document *PropSet, const Upnp_SID SubsId)

  *Similar to **UpnpAcceptSubscription()** except that it takes a DOM document for the variables to event rather than an array of strings.*

Control

- int UpnpSendAction (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Action, IXML_Document **RespNode)

  Sends a message to change a state variable in a service.
- int UpnpSendActionEx (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Header, IXML_Document *Action, IXML_Document **RespNode)

  Sends a message to change a state variable in a service.
- int UpnpSendActionAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Action, Upnp_FunPtr Fun, const void *Cookie)

  Sends a message to change a state variable in a service, generating a callback when the operation is complete.

- int UpnpSendActionExAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *ServiceType, const char *DevUDN, IXML_Document *Header, IXML_Document *Action, Upnp_FunPtr Fun, const void *Cookie)

  Sends a message to change a state variable in a service, generating a callback when the operation is complete.
- int UpnpGetServiceVarStatusAsync (UpnpClient_Handle Hnd, const char *ActionURL, const char *VarName, Upnp_FunPtr Fun, const void *Cookie)

  Queries the state of a variable of a service, generating a callback when the operation is complete.
- int UpnpGetServiceVarStatus (UpnpClient_Handle Hnd, const char *ActionURL, const char *VarName, DOMString *StVarVal)

  Queries the state of a state variable of a service on another device.

Control Point HTTP API

- int UpnpOpenHttpPost (const char *url, void **handle, const char *contentType, int contentLength, int timeout)

  Makes an HTTP POST request message, opens a connection to the server and sends the POST request to the server if the connection to the server succeeds.
- int UpnpWriteHttpPost (void *handle, char *buf, size_t *size, int timeout)

  *Sends a request to a server to copy the contents of a buffer to the URI specified in the **UpnpOpenHttpPost** call.*
- int UpnpCloseHttpPost (void *handle, int *httpStatus, int timeout)

  *Sends and receives any pending data, closes the connection with the server, and frees memory allocated during the **UpnpOpenHttpPost** call.*
- int UpnpOpenHttpGet (const char *url, void **handle, char **contentType, int *contentLength, int *httpStatus, int timeout)

  Gets a file specified in a URL.
- int UpnpOpenHttpGetProxy (const char *url, const char *proxy_str, void **handle, char **contentType, int *contentLength, int *httpStatus, int timeout)

  Gets a file specified in a URL through the specified proxy.
- int UpnpOpenHttpGetEx (const char *url, void **handle, char **contentType, int *contentLength, int *httpStatus, int lowRange, int highRange, int timeout)

  Gets specified number of bytes from a file specified in the URL.
- int UpnpCancelHttpGet (void *handle)

  *Set the cancel flag of the **handle** parameter.*
- int UpnpCloseHttpGet (void *handle)

  *Closes the connection and frees memory that was allocated for the **handle** parameter.*
- int UpnpReadHttpGet (void *handle, char *buf, size_t *size, int timeout)

  Gets specified number of bytes from a file specified in a URL.
- int UpnpHttpGetProgress (void *handle, size_t *length, size_t *total)

  Retrieve progress information of a http-get transfer.
- int UpnpOpenHttpConnection (const char *url, void **handle, int timeout)

  Opens a connection to the server.
- int UpnpMakeHttpRequest (Upnp_HttpMethod method, const char *url, void *handle, UpnpString *headers, const char *contentType, int contentLength, int timeout)

  *Makes a HTTP request using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpWriteHttpRequest (void *handle, char *buf, size_t *size, int timeout)

  *Writes the content of a HTTP request initiated by a **UpnpMakeHttpRequest** call. The end of the content should be indicated by a call to **UpnpEndHttpRequest**.*
- int UpnpEndHttpRequest (void *handle, int timeout)

  *Indicates the end of a HTTP request previously made by **UpnpMakeHttpRequest**.*
- int UpnpGetHttpResponse (void *handle, UpnpString *headers, char **contentType, int *contentLength, int *httpStatus, int timeout)

  *Gets the response from the server using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpReadHttpResponse (void *handle, char *buf, size_t *size, int timeout)

  *Reads the content of a response using a connection previously created by **UpnpOpenHttpConnection**.*
- int UpnpCloseHttpConnection (void *handle)

  *Closes the connection created with **UpnpOpenHttpConnection** and frees any memory associated with the connection.*
- int UpnpDownloadUrlItem (const char *url, char **outBuf, char *contentType)

Downloads a file specified in a URL.

- int UpnpDownloadXmlDoc (const char *url, IXML_Document **xmlDoc)

  Downloads an XML document specified in a URL.

Web Server API

- int UpnpAddVirtualDir (const char *dirName, const void *cookie, const void **oldcookie)

  Adds a virtual directory mapping.
- int UpnpRemoveVirtualDir (const char *dirName)

  *Removes a virtual directory mapping made with **UpnpAddVirtualDir**.*
- void UpnpRemoveAllVirtualDirs (void)

  Removes all virtual directory mappings.
- int UpnpEnableWebserver (int enable)

  Enables or disables the webserver.
- int UpnpIsWebserverEnabled (void)

  Returns `1` if the webserver is enabled, or `0` if it is not.
- void **UpnpSetHostValidateCallback** (WebCallback_HostValidate callback, void *cookie)
- void **UpnpSetAllowLiteralHostRedirection** (int enable)
- int UpnpVirtualDir_set_GetInfoCallback (VDCallback_GetInfo callback)

  Sets the get_info callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_OpenCallback (VDCallback_Open callback)

  Sets the open callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_ReadCallback (VDCallback_Read callback)

  Sets the read callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_WriteCallback (VDCallback_Write callback)

  Sets the write callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_SeekCallback (VDCallback_Seek callback)

  Sets the seek callback function to be used to access a virtual directory.
- int UpnpVirtualDir_set_CloseCallback (VDCallback_Close callback)

  Sets the close callback function to be used to access a virtual directory.

Variables

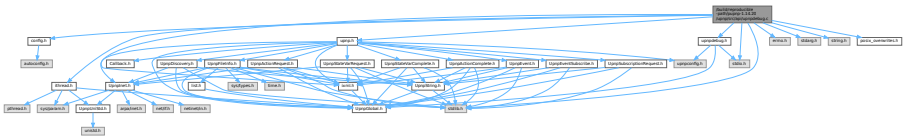
- struct VirtualDirCallbacks virtualDirCallback
- virtualDirList * pVirtualDirList
- ithread_mutex_t GlobalClientSubscribeMutex
- ithread_rwlock_t GlobalHndRWLock
- ithread_mutex_t gUUIDMutex
- ithread_mutex_t gSDKInitMutex = PTHREAD_MUTEX_INITIALIZER
- TimerThread gTimerThread
- ThreadPool gSendThreadPool
- ThreadPool gRecvThreadPool
- ThreadPool gMiniServerThreadPool
- WebServerState bWebServerState = WEB_SERVER_DISABLED
- WebCallback_HostValidate gWebCallback_HostValidate = 0
- void * gWebCallback_HostValidateCookie = 0
- int gAllowLiteralHostRedirection = 0
- char gIF_NAME [LINE_SIZE] = {'\0'}
- char gIF_IPV4 [INET_ADDRSTRLEN] = {'\0'}
- char gIF_IPV4_NETMASK [INET_ADDRSTRLEN] = {'\0'}
- char gIF_IPV6 [INET6_ADDRSTRLEN] = {'\0'}
- unsigned gIF_IPV6_PREFIX_LENGTH = 0
- char gIF_IPV6_ULA_GUA [INET6_ADDRSTRLEN] = {'\0'}
- unsigned gIF_IPV6_ULA_GUA_PREFIX_LENGTH = 0
- unsigned gIF_INDEX = (unsigned)-1

- unsigned short LOCAL_PORT_V4
- unsigned short LOCAL_PORT_V6
- unsigned short LOCAL_PORT_V6_ULA_GUA
- static void * HandleTable [NUM_HANDLE]
- membuffer gDocumentRootDir
- membuffer gWebserverCorsString
- size_t g_maxContentLength = DEFAULT_SOAP_CONTENT_LENGTH
- int g_UpnpSdkEQMaxLen = MAX_SUBSCRIPTION_QUEUED_EVENTS
- int g_UpnpSdkEQMaxAge = MAX_SUBSCRIPTION_EVENT_AGE
- int UpnpSdkInit = 0
- int UpnpSdkClientRegistered = 0
- int UpnpSdkDeviceRegisteredV4 = 0
- int UpnpSdkDeviceregisteredV6 = 0

9.76 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpdebug.c File Reference

```
#include "config.h"
#include "ithread.h"
#include "ixml.h"
#include "upnp.h"
#include "upnpdebug.h"
#include <errno.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for upnpdebug.c:



Functions

- int UpnpInitLog (void)

  Initialize the log files.
- void UpnpSetLogLevel (Upnp_LogLevel log_level)

  Set the log level (see `Upnp_LogLevel`).
- void UpnpCloseLog (void)

  Closes the log files.
- void UpnpSetLogFileNames (const char *newFileName, const char *ignored)

  Set the name for the log file. There used to be 2 separate files. The second parameter has been kept for compatibility but is ignored. Use a NULL file name for logging to stderr.
- static int **DebugAtThisLevel** (Upnp_LogLevel DLevel, Dbg_Module Module)
- static void **UpnpDisplayFileAndLine** (FILE *fp, const char *DbgFileName, int DbgLineNo, Upnp_LogLevel DLevel, Dbg_Module Module)

- void UpnpPrintf (Upnp_LogLevel DLevel, Dbg_Module Module, const char *DbgFileName, int DbgLineNo, const char *FmtStr,...)

  Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.
- FILE * UpnpGetDebugFile (Upnp_LogLevel DLevel, Dbg_Module Module)

  Check if the module is turned on for debug and returns the file descriptor corresponding to the debug level.

Variables

- static ithread_mutex_t GlobalDebugMutex
- static Upnp_LogLevel g_log_level = UPNP_DEFAULT_LOG_LEVEL
- static FILE * **fp**
- static int **is_stderr**
- static int **setlogwascalled**
- static int **initwascalled**
- static char * **fileName**

9.76.1 Function Documentation

9.76.1.1 UpnpCloseLog()

```
void UpnpCloseLog (
            void  )
```

Closes the log files.

References GlobalDebugMutex.

Referenced by UpnpFinish().

9.76.1.2 UpnpGetDebugFile()

```
FILE * UpnpGetDebugFile (
            Upnp_LogLevel level,
            Dbg_Module module )
```

Check if the module is turned on for debug and returns the file descriptor corresponding to the debug level.

Returns

NULL if the module is turn off for debug otherwise returns the right FILE pointer.

Parameters

in	*DLevel*	The level of the debug logging. It will decide whether debug statement will go to standard output, or any of the log files.
in	*Module*	debug will go in the name of this module.

9.76.1.3 UpnpInitLog()

```
int UpnpInitLog (
             void  )
```

Initialize the log files.

Returns

-1 if fails or UPNP_E_SUCCESS if succeeds.

References GlobalDebugMutex, and UPNP_E_SUCCESS.

Referenced by TvDeviceStart(), and UpnpInitPreamble().

9.76.1.4 UpnpPrintf()

```
void UpnpPrintf (
             Upnp_LogLevel DLevel,
             Dbg_Module Module,
             const char * DbgFileName,
             int DbgLineNo,
             const char * FmtStr,
              ...  )
```

Prints the debug statement either on the standard output or log file along with the information from where this debug statement is coming.

Parameters

in	*DLevel*	The level of the debug logging. It will decide whether debug statement will go to standard output, or any of the log files.
in	*Module*	debug will go in the name of this module.
in	*DbgFileName*	Name of the file from where debug statement is coming.
in	*DbgLineNo*	Line number of the file from where debug statement is coming.
in	*FmtStr*	Printf like format specification.
in	...	Printf like Variable number of arguments that will go in the debug statement.

References GlobalDebugMutex.

Referenced by AdvertiseAndReply(), AllocGenaHeaders(), create_ssdp_sock_reqv4(), create_ssdp_sock_v4(), DeviceAdvertisement(), DeviceShutdown(), FreeHandle(), gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_validate_delivery_urls(), GenaAutoRenewSubscription(), genaCallback(), genaInitNotify(), genaInitNotifyExt(), genaNotifyAll(), genaNotifyAllExt(), genaRenewSubscription(), genaSubscribe(), genaUnregisterDevice(), get_file_info(), get_miniserver_stopsock(), get_port(), GetHandleInfo(), handle_invoke_action(), handle_query_variable(), http_Connect(), http_MakeMessage(), http_ReadHttpResponse(), http_RecvMessage(), http_RecvPostMessage(), http_SendMessage(), NewRequestHandler(), notify_send_and_recv(), print_http_headers(), PrintHandleInfo(), printService(), printServiceList(), printServiceTable(), PrintThreadPoolStats(), readFromSSDPSocket(), RunMiniServer(), SearchByTarget(), send_action_response(), ServiceAdvertisement(), ServiceShutdown(), sock_destroy(), ssdp_handle_device_request(), start_event_handler(), StopMiniServer(), UpnpAcceptSubscription(), UpnpAcceptSubscriptionExt(), UpnpDownloadXmlDoc(), UpnpFinish(), UpnpGetIfInfo(), UpnpGetServiceVarStatus(), UpnpGetServiceVarStatusAsync(), UpnpInit2(), UpnpInitPreamble(), UpnpInitStartServers(), UpnpNotify(),

UpnpNotifyExt(), UpnpRegisterClient(), UpnpRegisterRootDevice(), UpnpRegisterRootDevice2(), UpnpRegisterRootDevice3(), UpnpRegisterRootDevice4(), UpnpRenewSubscription(), UpnpRenewSubscriptionAsync(), UpnpSearchAsync(), UpnpSendAction(), UpnpSendActionAsync(), UpnpSendActionEx(), UpnpSendActionExAsync(), UpnpSendAdvertisement(), UpnpSendAdvertisementLowPower(), UpnpSetMaxSubscriptions(), UpnpSetMaxSubscriptionTimeOut(), UpnpSubscribe(), UpnpSubscribeAsync(), UpnpThreadDistribution(), UpnpUnRegisterClient(), UpnpUnRegisterRootDevice(), UpnpUnRegisterRootDeviceLowPower(), UpnpUnSubscribe(), UpnpUnSubscribeAsync(), valid_ssdp_msg(), and web_server_callback().

9.76.1.5 UpnpSetLogFileNames()

```
void UpnpSetLogFileNames (
             const char * newFileName,
             const char * ignored )
```

Set the name for the log file. There used to be 2 separate files. The second parameter has been kept for compatibility but is ignored. Use a NULL file name for logging to stderr.

Parameters

in	*newFileName*	Name of the log file.
in	*ignored*	Ignored.

Referenced by TvDeviceStart().

9.76.1.6 UpnpSetLogLevel()

```
void UpnpSetLogLevel (
             Upnp_LogLevel log_level )
```

Set the log level (see `Upnp_LogLevel`).

Parameters

in	*log_level*	Log level.

References g_log_level.

Referenced by TvDeviceStart().

9.76.2 Variable Documentation

9.76.2.1 g_log_level

```
Upnp_LogLevel g_log_level = UPNP_DEFAULT_LOG_LEVEL  [static]
```

Global log level

Referenced by UpnpSetLogLevel().

9.76.2.2 GlobalDebugMutex

```
ithread_mutex_t GlobalDebugMutex  [static]
```

Mutex to synchronize all the log file operations in the debug mode

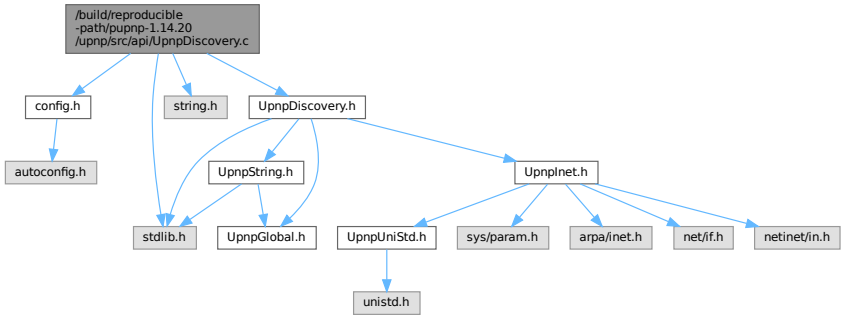
Referenced by UpnpCloseLog(), UpnpInitLog(), and UpnpPrintf().

9.77 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpDiscovery.c File Reference

Source file for UpnpDiscovery methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpDiscovery.h"
```

Include dependency graph for UpnpDiscovery.c:



Data Structures

- struct s_UpnpDiscovery

Functions

- UpnpDiscovery * UpnpDiscovery_new ()
- void UpnpDiscovery_delete (UpnpDiscovery *q)
- int UpnpDiscovery_assign (UpnpDiscovery *p, const UpnpDiscovery *q)
- UpnpDiscovery * UpnpDiscovery_dup (const UpnpDiscovery *q)
- int UpnpDiscovery_get_ErrCode (const UpnpDiscovery *p)
- int UpnpDiscovery_set_ErrCode (UpnpDiscovery *p, int n)
- int UpnpDiscovery_get_Expires (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Expires (UpnpDiscovery *p, int n)
- const UpnpString * UpnpDiscovery_get_DeviceID (const UpnpDiscovery *p)
- int UpnpDiscovery_set_DeviceID (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_DeviceID_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_DeviceID_cstr (const UpnpDiscovery *p)

- int UpnpDiscovery_strcpy_DeviceID (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_DeviceID (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_DeviceID (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_DeviceType (const UpnpDiscovery *p)
- int UpnpDiscovery_set_DeviceType (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_DeviceType_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_DeviceType_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_DeviceType (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_DeviceType (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_DeviceType (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_ServiceType (const UpnpDiscovery *p)
- int UpnpDiscovery_set_ServiceType (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_ServiceType_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_ServiceType_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_ServiceType (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_ServiceType (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_ServiceType (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_ServiceVer (const UpnpDiscovery *p)
- int UpnpDiscovery_set_ServiceVer (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_ServiceVer_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_ServiceVer_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_ServiceVer (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_ServiceVer (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_ServiceVer (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Location (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Location (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Location_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_Location_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Location (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Location (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Location (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Os (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Os (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Os_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_Os_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Os (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Os (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Os (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Date (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Date (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Date_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_Date_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Date (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Date (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Date (UpnpDiscovery *p)
- const UpnpString * UpnpDiscovery_get_Ext (const UpnpDiscovery *p)
- int UpnpDiscovery_set_Ext (UpnpDiscovery *p, const UpnpString *s)
- size_t UpnpDiscovery_get_Ext_Length (const UpnpDiscovery *p)
- const char * UpnpDiscovery_get_Ext_cstr (const UpnpDiscovery *p)
- int UpnpDiscovery_strcpy_Ext (UpnpDiscovery *p, const char *s)
- int UpnpDiscovery_strncpy_Ext (UpnpDiscovery *p, const char *s, size_t n)
- void UpnpDiscovery_clear_Ext (UpnpDiscovery *p)
- const struct sockaddr_storage * UpnpDiscovery_get_DestAddr (const UpnpDiscovery *p)
- int UpnpDiscovery_set_DestAddr (UpnpDiscovery *p, const struct sockaddr_storage *buf)
- void UpnpDiscovery_clear_DestAddr (UpnpDiscovery *p)

9.77.1 Detailed Description

Source file for UpnpDiscovery methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.77.2 Function Documentation

9.77.2.1 UpnpDiscovery_assign()

```
int UpnpDiscovery_assign (
            UpnpDiscovery * p,
            const UpnpDiscovery * q )
```

Assignment operator

References UpnpDiscovery_get_Date(), UpnpDiscovery_get_DestAddr(), UpnpDiscovery_get_DeviceID(), UpnpDiscovery_get_DeviceType(), UpnpDiscovery_get_ErrCode(), UpnpDiscovery_get_Expires(), UpnpDiscovery_get_Ext(), UpnpDiscovery_get_Location(), UpnpDiscovery_get_Os(), UpnpDiscovery_get_ServiceType(), UpnpDiscovery_get_ServiceVer(), UpnpDiscovery_set_Date(), UpnpDiscovery_set_DestAddr(), UpnpDiscovery_set_DeviceID(), UpnpDiscovery_set_DeviceType(), UpnpDiscovery_set_ErrCode(), UpnpDiscovery_set_Expires(), UpnpDiscovery_set_Ext(), UpnpDiscovery_set_Location(), UpnpDiscovery_set_Os(), UpnpDiscovery_set_ServiceType(), and UpnpDiscovery_set_ServiceVer().

Referenced by UpnpDiscovery_dup().

9.77.2.2 UpnpDiscovery_clear_Date()

```
void UpnpDiscovery_clear_Date (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Date

References UpnpString_clear().

9.77.2.3 UpnpDiscovery_clear_DestAddr()

```
void UpnpDiscovery_clear_DestAddr (
            UpnpDiscovery * p )
```

UpnpDiscovery_get_DestAddr

9.77.2.4 UpnpDiscovery_clear_DeviceID()

```
void UpnpDiscovery_clear_DeviceID (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_DeviceID

References UpnpString_clear().

9.77.2.5 UpnpDiscovery_clear_DeviceType()

```
void UpnpDiscovery_clear_DeviceType (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_DeviceType

References UpnpString_clear().

9.77.2.6 UpnpDiscovery_clear_Ext()

```
void UpnpDiscovery_clear_Ext (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Ext

References UpnpString_clear().

9.77.2.7 UpnpDiscovery_clear_Location()

```
void UpnpDiscovery_clear_Location (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Location

References UpnpString_clear().

9.77.2.8 UpnpDiscovery_clear_Os()

```
void UpnpDiscovery_clear_Os (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_Os

References UpnpString_clear().

9.77.2.9 UpnpDiscovery_clear_ServiceType()

```
void UpnpDiscovery_clear_ServiceType (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_ServiceType

References UpnpString_clear().

9.77.2.10 UpnpDiscovery_clear_ServiceVer()

```
void UpnpDiscovery_clear_ServiceVer (
            UpnpDiscovery * p )
```

UpnpDiscovery_clear_ServiceVer

References UpnpString_clear().

9.77.2.11 UpnpDiscovery_delete()

```
void UpnpDiscovery_delete (
            UpnpDiscovery * p )
```

Destructor

References UpnpString_delete().

Referenced by ssdp_handle_ctrlpt_msg(), SSDPResultData_delete(), and SSDPResultData_set_Param().

9.77.2.12 UpnpDiscovery_dup()

```
UpnpDiscovery * UpnpDiscovery_dup (
            const UpnpDiscovery * p )
```

Copy Constructor

References UpnpDiscovery_assign(), and UpnpDiscovery_new().

Referenced by SSDPResultData_set_Param().

9.77.2.13 UpnpDiscovery_get_Date()

```
const UpnpString * UpnpDiscovery_get_Date (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Date

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_Date_cstr(), and UpnpDiscovery_get_Date_Length().

9.77.2.14 UpnpDiscovery_get_Date_cstr()

```
const char * UpnpDiscovery_get_Date_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Date_cstr

References UpnpDiscovery_get_Date(), and UpnpString_get_String().

9.77.2.15 UpnpDiscovery_get_Date_Length()

```
size_t UpnpDiscovery_get_Date_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Date_Length

References UpnpDiscovery_get_Date(), and UpnpString_get_Length().

9.77.2.16 UpnpDiscovery_get_DestAddr()

```
const struct sockaddr_storage * UpnpDiscovery_get_DestAddr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DestAddr

Referenced by UpnpDiscovery_assign().

9.77.2.17 UpnpDiscovery_get_DeviceID()

```
const UpnpString * UpnpDiscovery_get_DeviceID (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceID

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_DeviceID_cstr(), and UpnpDiscovery_get_DeviceID_Length().

9.77.2.18 UpnpDiscovery_get_DeviceID_cstr()

```
const char * UpnpDiscovery_get_DeviceID_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceID_cstr

References UpnpDiscovery_get_DeviceID(), and UpnpString_get_String().

9.77.2.19 UpnpDiscovery_get_DeviceID_Length()

```
size_t UpnpDiscovery_get_DeviceID_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceID_Length

References UpnpDiscovery_get_DeviceID(), and UpnpString_get_Length().

9.77.2.20 UpnpDiscovery_get_DeviceType()

```
const UpnpString * UpnpDiscovery_get_DeviceType (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceType

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_DeviceType_cstr(), and UpnpDiscovery_get_DeviceType_Length().

9.77.2.21 UpnpDiscovery_get_DeviceType_cstr()

```
const char * UpnpDiscovery_get_DeviceType_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceType_cstr

References UpnpDiscovery_get_DeviceType(), and UpnpString_get_String().

9.77.2.22 UpnpDiscovery_get_DeviceType_Length()

```
size_t UpnpDiscovery_get_DeviceType_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_DeviceType_Length

References UpnpDiscovery_get_DeviceType(), and UpnpString_get_Length().

9.77.2.23 UpnpDiscovery_get_ErrCode()

```
int UpnpDiscovery_get_ErrCode (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpDiscovery_assign().

9.77.2.24 UpnpDiscovery_get_Expires()

```
int UpnpDiscovery_get_Expires (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Expires

Referenced by SampleUtil_PrintEvent(), ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.77.2.25 UpnpDiscovery_get_Ext()

```
const UpnpString * UpnpDiscovery_get_Ext (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Ext

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_Ext_cstr(), and UpnpDiscovery_get_Ext_Leng

9.77.2.26 UpnpDiscovery_get_Ext_cstr()

```
const char * UpnpDiscovery_get_Ext_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Ext_cstr

References UpnpDiscovery_get_Ext(), and UpnpString_get_String().

9.77.2.27 UpnpDiscovery_get_Ext_Length()

```
size_t UpnpDiscovery_get_Ext_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Ext_Length

References UpnpDiscovery_get_Ext(), and UpnpString_get_Length().

9.77.2.28 UpnpDiscovery_get_Location()

```
const UpnpString * UpnpDiscovery_get_Location (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Location

Referenced by SampleUtil_PrintEvent(), ssdp_handle_ctrlpt_msg(), UpnpDiscovery_assign(), UpnpDiscovery_get_Location_cstr(), and UpnpDiscovery_get_Location_Length().

9.77.2.29 UpnpDiscovery_get_Location_cstr()

```
const char * UpnpDiscovery_get_Location_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Location_cstr

References UpnpDiscovery_get_Location(), and UpnpString_get_String().

9.77.2.30 UpnpDiscovery_get_Location_Length()

```
size_t UpnpDiscovery_get_Location_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Location_Length

References UpnpDiscovery_get_Location(), and UpnpString_get_Length().

9.77.2.31 UpnpDiscovery_get_Os()

```
const UpnpString * UpnpDiscovery_get_Os (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Os

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_Os_cstr(), and UpnpDiscovery_get_Os_Lengt

9.77.2.32 UpnpDiscovery_get_Os_cstr()

```
const char * UpnpDiscovery_get_Os_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Os_cstr

References UpnpDiscovery_get_Os(), and UpnpString_get_String().

9.77.2.33 UpnpDiscovery_get_Os_Length()

```
size_t UpnpDiscovery_get_Os_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_Os_Length

References UpnpDiscovery_get_Os(), and UpnpString_get_Length().

9.77.2.34 UpnpDiscovery_get_ServiceType()

```
const UpnpString * UpnpDiscovery_get_ServiceType (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceType

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_ServiceType_cstr(), and UpnpDiscovery_get_ServiceType_Length().

9.77.2.35 UpnpDiscovery_get_ServiceType_cstr()

```
const char * UpnpDiscovery_get_ServiceType_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceType_cstr

References UpnpDiscovery_get_ServiceType(), and UpnpString_get_String().

9.77.2.36 UpnpDiscovery_get_ServiceType_Length()

```
size_t UpnpDiscovery_get_ServiceType_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceType_Length

References UpnpDiscovery_get_ServiceType(), and UpnpString_get_Length().

9.77.2.37 UpnpDiscovery_get_ServiceVer()

```
const UpnpString * UpnpDiscovery_get_ServiceVer (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceVer

Referenced by SampleUtil_PrintEvent(), UpnpDiscovery_assign(), UpnpDiscovery_get_ServiceVer_cstr(), and UpnpDiscovery_get_ServiceVer_Length().

9.77.2.38 UpnpDiscovery_get_ServiceVer_cstr()

```
const char * UpnpDiscovery_get_ServiceVer_cstr (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceVer_cstr

References UpnpDiscovery_get_ServiceVer(), and UpnpString_get_String().

9.77.2.39 UpnpDiscovery_get_ServiceVer_Length()

```
size_t UpnpDiscovery_get_ServiceVer_Length (
            const UpnpDiscovery * p )
```

UpnpDiscovery_get_ServiceVer_Length

References UpnpDiscovery_get_ServiceVer(), and UpnpString_get_Length().

9.77.2.40 UpnpDiscovery_new()

```
UpnpDiscovery * UpnpDiscovery_new ( )
```

Constructor

References UpnpString_new().

Referenced by ssdp_handle_ctrlpt_msg(), SSDPResultData_new(), and UpnpDiscovery_dup().

9.77.2.41 UpnpDiscovery_set_Date()

```
int UpnpDiscovery_set_Date (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Date

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.42 UpnpDiscovery_set_DestAddr()

```
int UpnpDiscovery_set_DestAddr (
            UpnpDiscovery * p,
            const struct sockaddr_storage * buf )
```

UpnpDiscovery_get_DestAddr

Referenced by ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.77.2.43 UpnpDiscovery_set_DeviceID()

```
int UpnpDiscovery_set_DeviceID (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_DeviceID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.44 UpnpDiscovery_set_DeviceType()

```
int UpnpDiscovery_set_DeviceType (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_DeviceType

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.45 UpnpDiscovery_set_ErrCode()

```
int UpnpDiscovery_set_ErrCode (
            UpnpDiscovery * p,
            int n )
```

UpnpDiscovery_set_ErrCode

Referenced by ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.77.2.46 UpnpDiscovery_set_Expires()

```
int UpnpDiscovery_set_Expires (
            UpnpDiscovery * p,
            int n )
```

UpnpDiscovery_set_Expires

Referenced by ssdp_handle_ctrlpt_msg(), and UpnpDiscovery_assign().

9.77.2.47 UpnpDiscovery_set_Ext()

```
int UpnpDiscovery_set_Ext (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Ext

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.48 UpnpDiscovery_set_Location()

```
int UpnpDiscovery_set_Location (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Location

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.49 UpnpDiscovery_set_Os()

```
int UpnpDiscovery_set_Os (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_Os

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.50 UpnpDiscovery_set_ServiceType()

```
int UpnpDiscovery_set_ServiceType (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_ServiceType

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.51 UpnpDiscovery_set_ServiceVer()

```
int UpnpDiscovery_set_ServiceVer (
            UpnpDiscovery * p,
            const UpnpString * s )
```

UpnpDiscovery_set_ServiceVer

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpDiscovery_assign().

9.77.2.52 UpnpDiscovery_strcpy_Date()

```
int UpnpDiscovery_strcpy_Date (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Date

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.77.2.53 UpnpDiscovery_strcpy_DeviceID()

```
int UpnpDiscovery_strcpy_DeviceID (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_DeviceID

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.77.2.54 UpnpDiscovery_strcpy_DeviceType()

```
int UpnpDiscovery_strcpy_DeviceType (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_DeviceType

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.77.2.55 UpnpDiscovery_strcpy_Ext()

```
int UpnpDiscovery_strcpy_Ext (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Ext

References UpnpString_set_String().

9.77.2.56 UpnpDiscovery_strcpy_Location()

```
int UpnpDiscovery_strcpy_Location (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Location

References UpnpString_set_String().

9.77.2.57 UpnpDiscovery_strcpy_Os()

```
int UpnpDiscovery_strcpy_Os (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_Os

References UpnpString_set_String().

9.77.2.58 UpnpDiscovery_strcpy_ServiceType()

```
int UpnpDiscovery_strcpy_ServiceType (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_ServiceType

References UpnpString_set_String().

Referenced by ssdp_handle_ctrlpt_msg().

9.77.2.59 UpnpDiscovery_strcpy_ServiceVer()

```
int UpnpDiscovery_strcpy_ServiceVer (
            UpnpDiscovery * p,
            const char * s )
```

UpnpDiscovery_strcpy_ServiceVer

References UpnpString_set_String().

9.77.2.60 UpnpDiscovery_strncpy_Date()

```
int UpnpDiscovery_strncpy_Date (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Date

References UpnpString_set_StringN().

9.77.2.61 UpnpDiscovery_strncpy_DeviceID()

```
int UpnpDiscovery_strncpy_DeviceID (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_DeviceID

References UpnpString_set_StringN().

9.77.2.62 UpnpDiscovery_strncpy_DeviceType()

```
int UpnpDiscovery_strncpy_DeviceType (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_DeviceType

References UpnpString_set_StringN().

9.77.2.63 UpnpDiscovery_strncpy_Ext()

```
int UpnpDiscovery_strncpy_Ext (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Ext

References UpnpString_set_StringN().

Referenced by ssdp_handle_ctrlpt_msg().

9.77.2.64 UpnpDiscovery_strncpy_Location()

```
int UpnpDiscovery_strncpy_Location (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Location

References UpnpString_set_StringN().

Referenced by ssdp_handle_ctrlpt_msg().

9.77.2.65 UpnpDiscovery_strncpy_Os()

```
int UpnpDiscovery_strncpy_Os (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_Os

References UpnpString_set_StringN().

Referenced by ssdp_handle_ctrlpt_msg().

9.77.2.66 UpnpDiscovery_strncpy_ServiceType()

```
int UpnpDiscovery_strncpy_ServiceType (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_ServiceType

References UpnpString_set_StringN().

9.77.2.67 UpnpDiscovery_strncpy_ServiceVer()

```
int UpnpDiscovery_strncpy_ServiceVer (
            UpnpDiscovery * p,
            const char * s,
            size_t n )
```

UpnpDiscovery_strncpy_ServiceVer

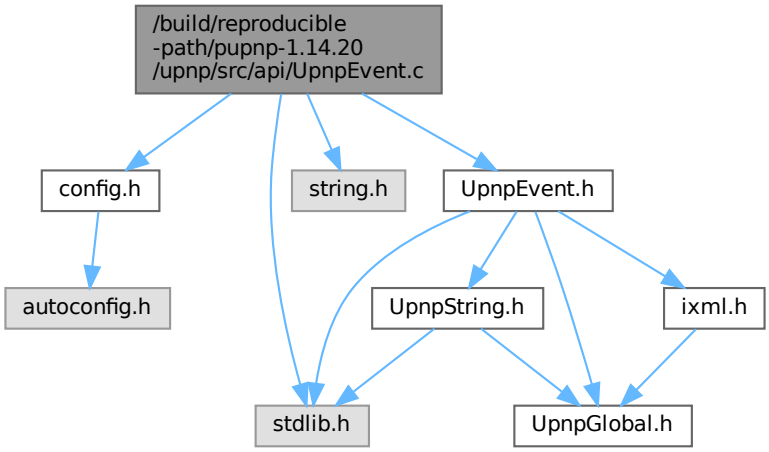
References UpnpString_set_StringN().

9.78 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEvent.c File Reference

Source file for UpnpEvent methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpEvent.h"
```

Include dependency graph for UpnpEvent.c:



Data Structures

- struct s_UpnpEvent

Functions

- UpnpEvent * UpnpEvent_new ()
- void UpnpEvent_delete (UpnpEvent *q)
- int UpnpEvent_assign (UpnpEvent *p, const UpnpEvent *q)
- UpnpEvent * UpnpEvent_dup (const UpnpEvent *q)
- int UpnpEvent_get_EventKey (const UpnpEvent *p)
- int UpnpEvent_set_EventKey (UpnpEvent *p, int n)
- IXML_Document * UpnpEvent_get_ChangedVariables (const UpnpEvent *p)
- int UpnpEvent_set_ChangedVariables (UpnpEvent *p, IXML_Document *n)
- const UpnpString * UpnpEvent_get_SID (const UpnpEvent *p)
- int UpnpEvent_set_SID (UpnpEvent *p, const UpnpString *s)
- size_t UpnpEvent_get_SID_Length (const UpnpEvent *p)
- const char * UpnpEvent_get_SID_cstr (const UpnpEvent *p)
- int UpnpEvent_strcpy_SID (UpnpEvent *p, const char *s)
- int UpnpEvent_strncpy_SID (UpnpEvent *p, const char *s, size_t n)
- void UpnpEvent_clear_SID (UpnpEvent *p)

9.78.1 Detailed Description

Source file for UpnpEvent methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.78.2 Function Documentation

9.78.2.1 UpnpEvent_assign()

```
int UpnpEvent_assign (
            UpnpEvent * p,
            const UpnpEvent * q )
```

Assignment operator

References UpnpEvent_get_ChangedVariables(), UpnpEvent_get_EventKey(), UpnpEvent_get_SID(), UpnpEvent_set_ChangedVaria
UpnpEvent_set_EventKey(), and UpnpEvent_set_SID().

Referenced by UpnpEvent_dup().

9.78.2.2 UpnpEvent_clear_SID()

```
void UpnpEvent_clear_SID (
            UpnpEvent * p )
```

UpnpEvent_clear_SID

References UpnpString_clear().

9.78.2.3 UpnpEvent_delete()

```
void UpnpEvent_delete (
            UpnpEvent * p )
```

Destructor

References UpnpString_delete().

Referenced by gena_process_notification_event().

9.78.2.4 UpnpEvent_dup()

```
UpnpEvent * UpnpEvent_dup (
            const UpnpEvent * p )
```

Copy Constructor

References UpnpEvent_assign(), and UpnpEvent_new().

9.78.2.5 UpnpEvent_get_ChangedVariables()

```
IXML_Document * UpnpEvent_get_ChangedVariables (
            const UpnpEvent * p )
```

UpnpEvent_get_ChangedVariables

Referenced by SampleUtil_PrintEvent(), and UpnpEvent_assign().

9.78.2.6 UpnpEvent_get_EventKey()

```
int UpnpEvent_get_EventKey (
            const UpnpEvent * p )
```

UpnpEvent_get_EventKey

Referenced by SampleUtil_PrintEvent(), and UpnpEvent_assign().

9.78.2.7 UpnpEvent_get_SID()

```
const UpnpString * UpnpEvent_get_SID (
            const UpnpEvent * p )
```

UpnpEvent_get_SID

Referenced by SampleUtil_PrintEvent(), UpnpEvent_assign(), UpnpEvent_get_SID_cstr(), and UpnpEvent_get_SID_Length().

9.78.2.8 UpnpEvent_get_SID_cstr()

```
const char * UpnpEvent_get_SID_cstr (
            const UpnpEvent * p )
```

UpnpEvent_get_SID_cstr

References UpnpEvent_get_SID(), and UpnpString_get_String().

9.78.2.9 UpnpEvent_get_SID_Length()

```
size_t UpnpEvent_get_SID_Length (
            const UpnpEvent * p )
```

UpnpEvent_get_SID_Length

References UpnpEvent_get_SID(), and UpnpString_get_Length().

9.78.2.10 UpnpEvent_new()

```
UpnpEvent * UpnpEvent_new ( )
```

Constructor

References UpnpString_new().

Referenced by gena_process_notification_event(), and UpnpEvent_dup().

9.78.2.11 UpnpEvent_set_ChangedVariables()

```
int UpnpEvent_set_ChangedVariables (
            UpnpEvent * p,
            IXML_Document * n )
```

UpnpEvent_set_ChangedVariables

Referenced by gena_process_notification_event(), and UpnpEvent_assign().

9.78.2.12 UpnpEvent_set_EventKey()

```
int UpnpEvent_set_EventKey (
            UpnpEvent * p,
            int n )
```

UpnpEvent_set_EventKey

Referenced by gena_process_notification_event(), and UpnpEvent_assign().

9.78.2.13 UpnpEvent_set_SID()

```
int UpnpEvent_set_SID (
            UpnpEvent * p,
            const UpnpString * s )
```

UpnpEvent_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by gena_process_notification_event(), and UpnpEvent_assign().

9.78.2.14 UpnpEvent_strcpy_SID()

```
int UpnpEvent_strcpy_SID (
            UpnpEvent * p,
            const char * s )
```

UpnpEvent_strcpy_SID

References UpnpString_set_String().

9.78.2.15 UpnpEvent_strncpy_SID()

```
int UpnpEvent_strncpy_SID (
            UpnpEvent * p,
            const char * s,
            size_t n )
```

UpnpEvent_strncpy_SID

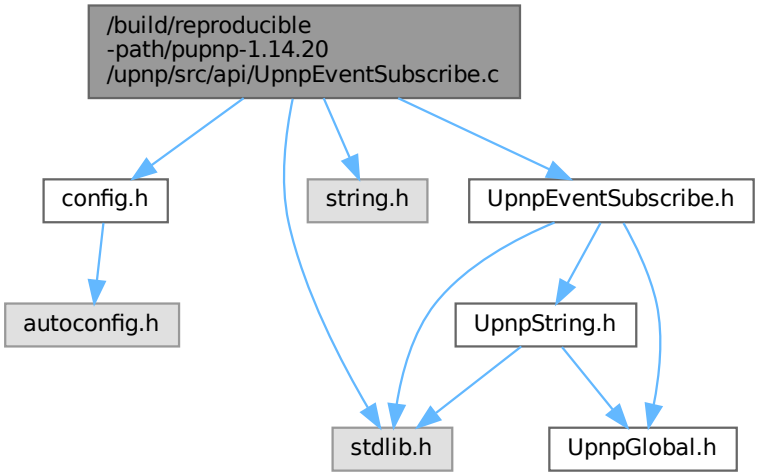
References UpnpString_set_StringN().

9.79 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEvent↩ Subscribe.c File Reference

Source file for UpnpEventSubscribe methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpEventSubscribe.h"
```

Include dependency graph for UpnpEventSubscribe.c:



Data Structures

- struct s_UpnpEventSubscribe

Functions

- UpnpEventSubscribe * UpnpEventSubscribe_new ()
- void UpnpEventSubscribe_delete (UpnpEventSubscribe *q)
- int UpnpEventSubscribe_assign (UpnpEventSubscribe *p, const UpnpEventSubscribe *q)
- UpnpEventSubscribe * UpnpEventSubscribe_dup (const UpnpEventSubscribe *q)
- int UpnpEventSubscribe_get_ErrCode (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_ErrCode (UpnpEventSubscribe *p, int n)
- int UpnpEventSubscribe_get_TimeOut (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_TimeOut (UpnpEventSubscribe *p, int n)
- const UpnpString * UpnpEventSubscribe_get_SID (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_SID (UpnpEventSubscribe *p, const UpnpString *s)
- size_t UpnpEventSubscribe_get_SID_Length (const UpnpEventSubscribe *p)
- const char * UpnpEventSubscribe_get_SID_cstr (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_strcpy_SID (UpnpEventSubscribe *p, const char *s)
- int UpnpEventSubscribe_strncpy_SID (UpnpEventSubscribe *p, const char *s, size_t n)
- void UpnpEventSubscribe_clear_SID (UpnpEventSubscribe *p)
- const UpnpString * UpnpEventSubscribe_get_PublisherUrl (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_set_PublisherUrl (UpnpEventSubscribe *p, const UpnpString *s)
- size_t UpnpEventSubscribe_get_PublisherUrl_Length (const UpnpEventSubscribe *p)
- const char * UpnpEventSubscribe_get_PublisherUrl_cstr (const UpnpEventSubscribe *p)
- int UpnpEventSubscribe_strcpy_PublisherUrl (UpnpEventSubscribe *p, const char *s)
- int UpnpEventSubscribe_strncpy_PublisherUrl (UpnpEventSubscribe *p, const char *s, size_t n)
- void UpnpEventSubscribe_clear_PublisherUrl (UpnpEventSubscribe *p)

9.79.1 Detailed Description

Source file for UpnpEventSubscribe methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.79.2 Function Documentation

9.79.2.1 UpnpEventSubscribe_assign()

```
int UpnpEventSubscribe_assign (
            UpnpEventSubscribe * p,
            const UpnpEventSubscribe * q )
```

Assignment operator

References UpnpEventSubscribe_get_ErrCode(), UpnpEventSubscribe_get_PublisherUrl(), UpnpEventSubscribe_get_SID(), UpnpEventSubscribe_get_TimeOut(), UpnpEventSubscribe_set_ErrCode(), UpnpEventSubscribe_set_PublisherUrl(), UpnpEventSubscribe_set_SID(), and UpnpEventSubscribe_set_TimeOut().

Referenced by UpnpEventSubscribe_dup().

9.79.2.2 UpnpEventSubscribe_clear_PublisherUrl()

```
void UpnpEventSubscribe_clear_PublisherUrl (
            UpnpEventSubscribe * p )
```

UpnpEventSubscribe_clear_PublisherUrl

References UpnpString_clear().

9.79.2.3 UpnpEventSubscribe_clear_SID()

```
void UpnpEventSubscribe_clear_SID (
            UpnpEventSubscribe * p )
```

UpnpEventSubscribe_clear_SID

References UpnpString_clear().

9.79.2.4 UpnpEventSubscribe_delete()

```
void UpnpEventSubscribe_delete (
            UpnpEventSubscribe * p )
```

Destructor

References UpnpString_delete().

Referenced by free_subscribe_arg(), and UpnpThreadDistribution().

9.79.2.5 UpnpEventSubscribe_dup()

```
UpnpEventSubscribe * UpnpEventSubscribe_dup (
            const UpnpEventSubscribe * p )
```

Copy Constructor

References UpnpEventSubscribe_assign(), and UpnpEventSubscribe_new().

9.79.2.6 UpnpEventSubscribe_get_ErrCode()

```
int UpnpEventSubscribe_get_ErrCode (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpEventSubscribe_assign().

9.79.2.7 UpnpEventSubscribe_get_PublisherUrl()

```
const UpnpString * UpnpEventSubscribe_get_PublisherUrl (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_PublisherUrl

Referenced by SampleUtil_PrintEvent(), UpnpEventSubscribe_assign(), UpnpEventSubscribe_get_PublisherUrl_cstr(), UpnpEventSubscribe_get_PublisherUrl_Length(), and UpnpThreadDistribution().

9.79.2.8 UpnpEventSubscribe_get_PublisherUrl_cstr()

```
const char * UpnpEventSubscribe_get_PublisherUrl_cstr (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_PublisherUrl_cstr

References UpnpEventSubscribe_get_PublisherUrl(), and UpnpString_get_String().

9.79.2.9 UpnpEventSubscribe_get_PublisherUrl_Length()

```
size_t UpnpEventSubscribe_get_PublisherUrl_Length (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_PublisherUrl_Length

References UpnpEventSubscribe_get_PublisherUrl(), and UpnpString_get_Length().

9.79.2.10 UpnpEventSubscribe_get_SID()

```
const UpnpString * UpnpEventSubscribe_get_SID (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_SID

Referenced by GenaAutoRenewSubscription(), SampleUtil_PrintEvent(), UpnpEventSubscribe_assign(), UpnpEventSubscribe_get_SID_cstr(), UpnpEventSubscribe_get_SID_Length(), and UpnpThreadDistribution().

9.79.2.11 UpnpEventSubscribe_get_SID_cstr()

```
const char * UpnpEventSubscribe_get_SID_cstr (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_SID_cstr

References UpnpEventSubscribe_get_SID(), and UpnpString_get_String().

9.79.2.12 UpnpEventSubscribe_get_SID_Length()

```
size_t UpnpEventSubscribe_get_SID_Length (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_SID_Length

References UpnpEventSubscribe_get_SID(), and UpnpString_get_Length().

9.79.2.13 UpnpEventSubscribe_get_TimeOut()

```
int UpnpEventSubscribe_get_TimeOut (
            const UpnpEventSubscribe * p )
```

UpnpEventSubscribe_get_TimeOut

Referenced by GenaAutoRenewSubscription(), SampleUtil_PrintEvent(), and UpnpEventSubscribe_assign().

9.79.2.14 UpnpEventSubscribe_new()

```
UpnpEventSubscribe * UpnpEventSubscribe_new ( )
```

Constructor

References UpnpString_new().

Referenced by ScheduleGenaAutoRenew(), UpnpEventSubscribe_dup(), and UpnpThreadDistribution().

9.79.2.15 UpnpEventSubscribe_set_ErrCode()

```
int UpnpEventSubscribe_set_ErrCode (
            UpnpEventSubscribe * p,
            int n )
```

UpnpEventSubscribe_set_ErrCode

Referenced by GenaAutoRenewSubscription(), ScheduleGenaAutoRenew(), UpnpEventSubscribe_assign(), and UpnpThreadDistribution().

9.79.2.16 UpnpEventSubscribe_set_PublisherUrl()

```
int UpnpEventSubscribe_set_PublisherUrl (
            UpnpEventSubscribe * p,
            const UpnpString * s )
```

UpnpEventSubscribe_set_PublisherUrl

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by ScheduleGenaAutoRenew(), and UpnpEventSubscribe_assign().

9.79.2.17 UpnpEventSubscribe_set_SID()

```
int UpnpEventSubscribe_set_SID (
            UpnpEventSubscribe * p,
            const UpnpString * s )
```

UpnpEventSubscribe_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by ScheduleGenaAutoRenew(), UpnpEventSubscribe_assign(), and UpnpThreadDistribution().

9.79.2.18 UpnpEventSubscribe_set_TimeOut()

```
int UpnpEventSubscribe_set_TimeOut (
            UpnpEventSubscribe * p,
            int n )
```

UpnpEventSubscribe_set_TimeOut

Referenced by GenaAutoRenewSubscription(), ScheduleGenaAutoRenew(), UpnpEventSubscribe_assign(), and UpnpThreadDistribution().

9.79.2.19 UpnpEventSubscribe_strcpy_PublisherUrl()

```
int UpnpEventSubscribe_strcpy_PublisherUrl (
            UpnpEventSubscribe * p,
            const char * s )
```

UpnpEventSubscribe_strcpy_PublisherUrl

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.79.2.20 UpnpEventSubscribe_strcpy_SID()

```
int UpnpEventSubscribe_strcpy_SID (
            UpnpEventSubscribe * p,
            const char * s )
```

UpnpEventSubscribe_strcpy_SID

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.79.2.21 UpnpEventSubscribe_strncpy_PublisherUrl()

```
int UpnpEventSubscribe_strncpy_PublisherUrl (
             UpnpEventSubscribe * p,
             const char * s,
             size_t n )
```

UpnpEventSubscribe_strncpy_PublisherUrl

References UpnpString_set_StringN().

9.79.2.22 UpnpEventSubscribe_strncpy_SID()

```
int UpnpEventSubscribe_strncpy_SID (
             UpnpEventSubscribe * p,
             const char * s,
             size_t n )
```

UpnpEventSubscribe_strncpy_SID

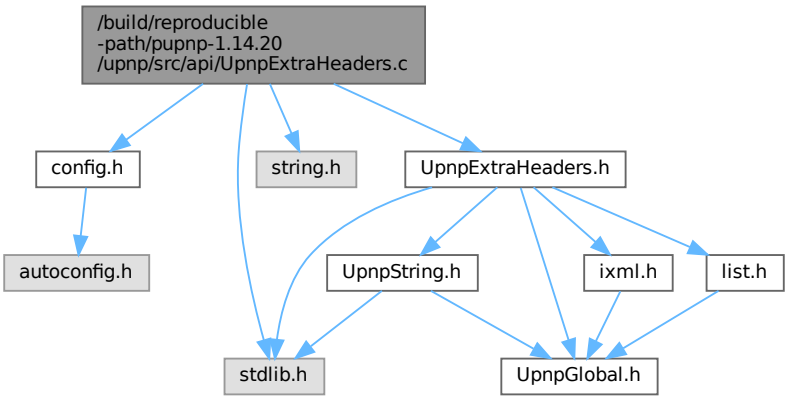
References UpnpString_set_StringN().

9.80 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpExtra↩ Headers.c File Reference

Source file for UpnpExtraHeaders methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpExtraHeaders.h"
```

Include dependency graph for UpnpExtraHeaders.c:



Data Structures

- struct s_UpnpExtraHeaders

Functions

- UpnpExtraHeaders * UpnpExtraHeaders_new ()
- void UpnpExtraHeaders_delete (UpnpExtraHeaders *q)
- int UpnpExtraHeaders_assign (UpnpExtraHeaders *p, const UpnpExtraHeaders *q)
- UpnpExtraHeaders * UpnpExtraHeaders_dup (const UpnpExtraHeaders *q)
- const UpnpListHead * UpnpExtraHeaders_get_node (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_node (UpnpExtraHeaders *p, const UpnpListHead *q)
- void UpnpExtraHeaders_add_to_list_node (UpnpExtraHeaders *p, struct UpnpListHead *head)
- const UpnpString * UpnpExtraHeaders_get_name (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_name (UpnpExtraHeaders *p, const UpnpString *s)
- size_t UpnpExtraHeaders_get_name_Length (const UpnpExtraHeaders *p)
- const char * UpnpExtraHeaders_get_name_cstr (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_strcpy_name (UpnpExtraHeaders *p, const char *s)
- int UpnpExtraHeaders_strncpy_name (UpnpExtraHeaders *p, const char *s, size_t n)
- void UpnpExtraHeaders_clear_name (UpnpExtraHeaders *p)
- const UpnpString * UpnpExtraHeaders_get_value (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_value (UpnpExtraHeaders *p, const UpnpString *s)
- size_t UpnpExtraHeaders_get_value_Length (const UpnpExtraHeaders *p)
- const char * UpnpExtraHeaders_get_value_cstr (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_strcpy_value (UpnpExtraHeaders *p, const char *s)
- int UpnpExtraHeaders_strncpy_value (UpnpExtraHeaders *p, const char *s, size_t n)
- void UpnpExtraHeaders_clear_value (UpnpExtraHeaders *p)
- const DOMString UpnpExtraHeaders_get_resp (const UpnpExtraHeaders *p)
- int UpnpExtraHeaders_set_resp (UpnpExtraHeaders *p, const DOMString s)
- const char * UpnpExtraHeaders_get_resp_cstr (const UpnpExtraHeaders *p)

9.80.1 Detailed Description

Source file for UpnpExtraHeaders methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.80.2 Function Documentation

9.80.2.1 UpnpExtraHeaders_add_to_list_node()

```
void UpnpExtraHeaders_add_to_list_node (
            UpnpExtraHeaders * p,
            UpnpListHead * head )
```

UpnpExtraHeaders_add_to_list_node

9.80.2.2 UpnpExtraHeaders_assign()

```
int UpnpExtraHeaders_assign (
            UpnpExtraHeaders * p,
            const UpnpExtraHeaders * q )
```

Assignment operator

References UpnpExtraHeaders_get_name(), UpnpExtraHeaders_get_node(), UpnpExtraHeaders_get_resp(), UpnpExtraHeaders_get_value(), UpnpExtraHeaders_set_name(), UpnpExtraHeaders_set_node(), UpnpExtraHeaders_set_resp(), and UpnpExtraHeaders_set_value().

Referenced by UpnpExtraHeaders_dup().

9.80.2.3 UpnpExtraHeaders_clear_name()

```
void UpnpExtraHeaders_clear_name (
            UpnpExtraHeaders * p )
```

UpnpExtraHeaders_clear_name

References UpnpString_clear().

9.80.2.4 UpnpExtraHeaders_clear_value()

```
void UpnpExtraHeaders_clear_value (
            UpnpExtraHeaders * p )
```

UpnpExtraHeaders_clear_value

References UpnpString_clear().

9.80.2.5 UpnpExtraHeaders_delete()

```
void UpnpExtraHeaders_delete (
            UpnpExtraHeaders * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by FreeExtraHTTPHeaders().

9.80.2.6 UpnpExtraHeaders_dup()

```
UpnpExtraHeaders * UpnpExtraHeaders_dup (
            const UpnpExtraHeaders * p )
```

Copy Constructor

References UpnpExtraHeaders_assign(), and UpnpExtraHeaders_new().

9.80.2.7 UpnpExtraHeaders_get_name()

```
const UpnpString * UpnpExtraHeaders_get_name (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_name

Referenced by UpnpExtraHeaders_assign(), UpnpExtraHeaders_get_name_cstr(), and UpnpExtraHeaders_get_name_Length().

9.80.2.8 UpnpExtraHeaders_get_name_cstr()

```
const char * UpnpExtraHeaders_get_name_cstr (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_name_cstr

References UpnpExtraHeaders_get_name(), and UpnpString_get_String().

9.80.2.9 UpnpExtraHeaders_get_name_Length()

```
size_t UpnpExtraHeaders_get_name_Length (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_name_Length

References UpnpExtraHeaders_get_name(), and UpnpString_get_Length().

9.80.2.10 UpnpExtraHeaders_get_node()

```
const UpnpListHead * UpnpExtraHeaders_get_node (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_node

Referenced by ExtraHTTPHeaders(), and UpnpExtraHeaders_assign().

9.80.2.11 UpnpExtraHeaders_get_resp()

```
const DOMString UpnpExtraHeaders_get_resp (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_resp

Referenced by http_MakeMessage(), UpnpExtraHeaders_assign(), and UpnpExtraHeaders_get_resp_cstr().

9.80.2.12 UpnpExtraHeaders_get_resp_cstr()

```
const char * UpnpExtraHeaders_get_resp_cstr (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_resp_cstr

References UpnpExtraHeaders_get_resp().

9.80.2.13 UpnpExtraHeaders_get_value()

```
const UpnpString * UpnpExtraHeaders_get_value (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_value

Referenced by UpnpExtraHeaders_assign(), UpnpExtraHeaders_get_value_cstr(), and UpnpExtraHeaders_get_value_Length().

9.80.2.14 UpnpExtraHeaders_get_value_cstr()

```
const char * UpnpExtraHeaders_get_value_cstr (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_value_cstr

References UpnpExtraHeaders_get_value(), and UpnpString_get_String().

9.80.2.15 UpnpExtraHeaders_get_value_Length()

```
size_t UpnpExtraHeaders_get_value_Length (
            const UpnpExtraHeaders * p )
```

UpnpExtraHeaders_get_value_Length

References UpnpExtraHeaders_get_value(), and UpnpString_get_Length().

9.80.2.16 UpnpExtraHeaders_new()

```
UpnpExtraHeaders * UpnpExtraHeaders_new ( )
```

Constructor

References UpnpString_new().

Referenced by ExtraHTTPHeaders(), and UpnpExtraHeaders_dup().

9.80.2.17 UpnpExtraHeaders_set_name()

```
int UpnpExtraHeaders_set_name (
            UpnpExtraHeaders * p,
            const UpnpString * s )
```

UpnpExtraHeaders_set_name

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpExtraHeaders_assign().

9.80.2.18 UpnpExtraHeaders_set_node()

```
int UpnpExtraHeaders_set_node (
            UpnpExtraHeaders * p,
            const UpnpListHead * q )
```

UpnpExtraHeaders_set_node

Referenced by UpnpExtraHeaders_assign().

9.80.2.19 UpnpExtraHeaders_set_resp()

```
int UpnpExtraHeaders_set_resp (
            UpnpExtraHeaders * p,
            const DOMString s )
```

UpnpExtraHeaders_set_resp

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by UpnpExtraHeaders_assign().

9.80.2.20 UpnpExtraHeaders_set_value()

```
int UpnpExtraHeaders_set_value (
            UpnpExtraHeaders * p,
            const UpnpString * s )
```

UpnpExtraHeaders_set_value

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpExtraHeaders_assign().

9.80.2.21 UpnpExtraHeaders_strcpy_name()

```
int UpnpExtraHeaders_strcpy_name (
            UpnpExtraHeaders * p,
            const char * s )
```

UpnpExtraHeaders_strcpy_name

References UpnpString_set_String().

9.80.2.22 UpnpExtraHeaders_strcpy_value()

```
int UpnpExtraHeaders_strcpy_value (
            UpnpExtraHeaders * p,
            const char * s )
```

UpnpExtraHeaders_strcpy_value

References UpnpString_set_String().

9.80.2.23 UpnpExtraHeaders_strncpy_name()

```
int UpnpExtraHeaders_strncpy_name (
            UpnpExtraHeaders * p,
            const char * s,
            size_t n )
```

UpnpExtraHeaders_strncpy_name

References UpnpString_set_StringN().

Referenced by ExtraHTTPHeaders().

9.80.2.24 UpnpExtraHeaders_strncpy_value()

```
int UpnpExtraHeaders_strncpy_value (
            UpnpExtraHeaders * p,
            const char * s,
            size_t n )
```

UpnpExtraHeaders_strncpy_value

References UpnpString_set_StringN().

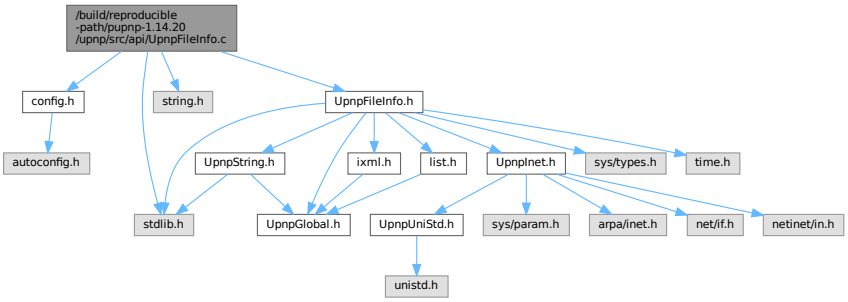
Referenced by ExtraHTTPHeaders().

9.81 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpFileInfo.c File Reference

Source file for UpnpFileInfo methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpFileInfo.h"
```

Include dependency graph for UpnpFileInfo.c:



Data Structures

- struct s_UpnpFileInfo

Functions

- UpnpFileInfo * UpnpFileInfo_new ()
- void UpnpFileInfo_delete (UpnpFileInfo *q)
- int UpnpFileInfo_assign (UpnpFileInfo *p, const UpnpFileInfo *q)
- UpnpFileInfo * UpnpFileInfo_dup (const UpnpFileInfo *q)
- off_t UpnpFileInfo_get_FileLength (const UpnpFileInfo *p)
- int UpnpFileInfo_set_FileLength (UpnpFileInfo *p, off_t n)
- time_t UpnpFileInfo_get_LastModified (const UpnpFileInfo *p)
- int UpnpFileInfo_set_LastModified (UpnpFileInfo *p, time_t n)
- int UpnpFileInfo_get_IsDirectory (const UpnpFileInfo *p)
- int UpnpFileInfo_set_IsDirectory (UpnpFileInfo *p, int n)
- int UpnpFileInfo_get_IsReadable (const UpnpFileInfo *p)
- int UpnpFileInfo_set_IsReadable (UpnpFileInfo *p, int n)
- const DOMString UpnpFileInfo_get_ContentType (const UpnpFileInfo *p)
- int UpnpFileInfo_set_ContentType (UpnpFileInfo *p, const DOMString s)
- const char * UpnpFileInfo_get_ContentType_cstr (const UpnpFileInfo *p)
- const UpnpListHead * UpnpFileInfo_get_ExtraHeadersList (const UpnpFileInfo *p)
- int UpnpFileInfo_set_ExtraHeadersList (UpnpFileInfo *p, const UpnpListHead *q)
- void UpnpFileInfo_add_to_list_ExtraHeadersList (UpnpFileInfo *p, struct UpnpListHead *head)
- const struct sockaddr_storage * UpnpFileInfo_get_CtrlPtIPAddr (const UpnpFileInfo *p)
- int UpnpFileInfo_set_CtrlPtIPAddr (UpnpFileInfo *p, const struct sockaddr_storage *buf)
- void UpnpFileInfo_clear_CtrlPtIPAddr (UpnpFileInfo *p)
- const UpnpString * UpnpFileInfo_get_Os (const UpnpFileInfo *p)
- int UpnpFileInfo_set_Os (UpnpFileInfo *p, const UpnpString *s)
- size_t UpnpFileInfo_get_Os_Length (const UpnpFileInfo *p)
- const char * UpnpFileInfo_get_Os_cstr (const UpnpFileInfo *p)
- int UpnpFileInfo_strcpy_Os (UpnpFileInfo *p, const char *s)
- int UpnpFileInfo_strncpy_Os (UpnpFileInfo *p, const char *s, size_t n)
- void UpnpFileInfo_clear_Os (UpnpFileInfo *p)

9.81.1 Detailed Description

Source file for UpnpFileInfo methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.81.2 Function Documentation

9.81.2.1 UpnpFileInfo_add_to_list_ExtraHeadersList()

```
void UpnpFileInfo_add_to_list_ExtraHeadersList (
            UpnpFileInfo * p,
            UpnpListHead * head )
```

UpnpFileInfo_add_to_list_ExtraHeadersList

9.81.2.2 UpnpFileInfo_assign()

```
int UpnpFileInfo_assign (
            UpnpFileInfo * p,
            const UpnpFileInfo * q )
```

Assignment operator

References UpnpFileInfo_get_ContentType(), UpnpFileInfo_get_CtrlPtIPAddr(), UpnpFileInfo_get_ExtraHeadersList(), UpnpFileInfo_get_FileLength(), UpnpFileInfo_get_IsDirectory(), UpnpFileInfo_get_IsReadable(), UpnpFileInfo_get_LastModified(), UpnpFileInfo_get_Os(), UpnpFileInfo_set_ContentType(), UpnpFileInfo_set_CtrlPtIPAddr(), UpnpFileInfo_set_ExtraHeadersList(), UpnpFileInfo_set_FileLength(), UpnpFileInfo_set_IsDirectory(), UpnpFileInfo_set_IsReadable(), UpnpFileInfo_set_LastModified(), and UpnpFileInfo_set_Os().

Referenced by UpnpFileInfo_dup().

9.81.2.3 UpnpFileInfo_clear_CtrlPtIPAddr()

```
void UpnpFileInfo_clear_CtrlPtIPAddr (
            UpnpFileInfo * p )
```

UpnpFileInfo_get_CtrlPtIPAddr

9.81.2.4 UpnpFileInfo_clear_Os()

```
void UpnpFileInfo_clear_Os (
            UpnpFileInfo * p )
```

UpnpFileInfo_clear_Os

References UpnpString_clear().

9.81.2.5 UpnpFileInfo_delete()

```
void UpnpFileInfo_delete (
            UpnpFileInfo * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by process_request().

9.81.2.6 UpnpFileInfo_dup()

```
UpnpFileInfo * UpnpFileInfo_dup (
            const UpnpFileInfo * p )
```

Copy Constructor

References UpnpFileInfo_assign(), and UpnpFileInfo_new().

9.81.2.7 UpnpFileInfo_get_ContentType()

```
const DOMString UpnpFileInfo_get_ContentType (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_ContentType

Referenced by get_content_type(), process_request(), UpnpFileInfo_assign(), and UpnpFileInfo_get_ContentType_cstr().

9.81.2.8 UpnpFileInfo_get_ContentType_cstr()

```
const char * UpnpFileInfo_get_ContentType_cstr (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_ContentType_cstr

References UpnpFileInfo_get_ContentType().

9.81.2.9 UpnpFileInfo_get_CtrlPtIPAddr()

```
const struct sockaddr_storage * UpnpFileInfo_get_CtrlPtIPAddr (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_CtrlPtIPAddr

Referenced by UpnpFileInfo_assign().

9.81.2.10 UpnpFileInfo_get_ExtraHeadersList()

```
const UpnpListHead * UpnpFileInfo_get_ExtraHeadersList (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_ExtraHeadersList

Referenced by process_request(), and UpnpFileInfo_assign().

9.81.2.11 UpnpFileInfo_get_FileLength()

```
off_t UpnpFileInfo_get_FileLength (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_FileLength

Referenced by get_file_info(), process_request(), and UpnpFileInfo_assign().

9.81.2.12 UpnpFileInfo_get_IsDirectory()

```
int UpnpFileInfo_get_IsDirectory (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_IsDirectory

Referenced by process_request(), and UpnpFileInfo_assign().

9.81.2.13 UpnpFileInfo_get_IsReadable()

```
int UpnpFileInfo_get_IsReadable (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_IsReadable

Referenced by get_file_info(), process_request(), and UpnpFileInfo_assign().

9.81.2.14 UpnpFileInfo_get_LastModified()

```
time_t UpnpFileInfo_get_LastModified (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_LastModified

Referenced by get_file_info(), process_request(), and UpnpFileInfo_assign().

9.81.2.15 UpnpFileInfo_get_Os()

```
const UpnpString * UpnpFileInfo_get_Os (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_Os

Referenced by UpnpFileInfo_assign(), UpnpFileInfo_get_Os_cstr(), and UpnpFileInfo_get_Os_Length().

9.81.2.16 UpnpFileInfo_get_Os_cstr()

```
const char * UpnpFileInfo_get_Os_cstr (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_Os_cstr

References UpnpFileInfo_get_Os(), and UpnpString_get_String().

9.81.2.17 UpnpFileInfo_get_Os_Length()

```
size_t UpnpFileInfo_get_Os_Length (
            const UpnpFileInfo * p )
```

UpnpFileInfo_get_Os_Length

References UpnpFileInfo_get_Os(), and UpnpString_get_Length().

9.81.2.18 UpnpFileInfo_new()

```
UpnpFileInfo * UpnpFileInfo_new ( )
```

Constructor

References UpnpString_new().

Referenced by process_request(), and UpnpFileInfo_dup().

9.81.2.19 UpnpFileInfo_set_ContentType()

```
int UpnpFileInfo_set_ContentType (
            UpnpFileInfo * p,
            const DOMString s )
```

UpnpFileInfo_set_ContentType

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by get_content_type(), get_file_info(), process_request(), and UpnpFileInfo_assign().

9.81.2.20 UpnpFileInfo_set_CtrlPtIPAddr()

```
int UpnpFileInfo_set_CtrlPtIPAddr (
            UpnpFileInfo * p,
            const struct sockaddr_storage * buf )
```

UpnpFileInfo_get_CtrlPtIPAddr

Referenced by process_request(), and UpnpFileInfo_assign().

9.81.2.21 UpnpFileInfo_set_ExtraHeadersList()

```
int UpnpFileInfo_set_ExtraHeadersList (
            UpnpFileInfo * p,
            const UpnpListHead * q )
```

UpnpFileInfo_set_ExtraHeadersList

Referenced by UpnpFileInfo_assign().

9.81.2.22 UpnpFileInfo_set_FileLength()

```
int UpnpFileInfo_set_FileLength (
            UpnpFileInfo * p,
            off_t n )
```

UpnpFileInfo_set_FileLength

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.81.2.23 UpnpFileInfo_set_IsDirectory()

```
int UpnpFileInfo_set_IsDirectory (
            UpnpFileInfo * p,
            int n )
```

UpnpFileInfo_set_IsDirectory

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.81.2.24 UpnpFileInfo_set_IsReadable()

```
int UpnpFileInfo_set_IsReadable (
            UpnpFileInfo * p,
            int n )
```

UpnpFileInfo_set_IsReadable

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.81.2.25 UpnpFileInfo_set_LastModified()

```
int UpnpFileInfo_set_LastModified (
            UpnpFileInfo * p,
            time_t n )
```

UpnpFileInfo_set_LastModified

Referenced by get_alias(), get_file_info(), and UpnpFileInfo_assign().

9.81.2.26 UpnpFileInfo_set_Os()

```
int UpnpFileInfo_set_Os (
            UpnpFileInfo * p,
            const UpnpString * s )
```

UpnpFileInfo_set_Os

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpFileInfo_assign().

9.81.2.27 UpnpFileInfo_strcpy_Os()

```
int UpnpFileInfo_strcpy_Os (
            UpnpFileInfo * p,
            const char * s )
```

UpnpFileInfo_strcpy_Os

References UpnpString_set_String().

9.81.2.28 UpnpFileInfo_strncpy_Os()

```
int UpnpFileInfo_strncpy_Os (
            UpnpFileInfo * p,
            const char * s,
            size_t n )
```

UpnpFileInfo_strncpy_Os

References UpnpString_set_StringN().

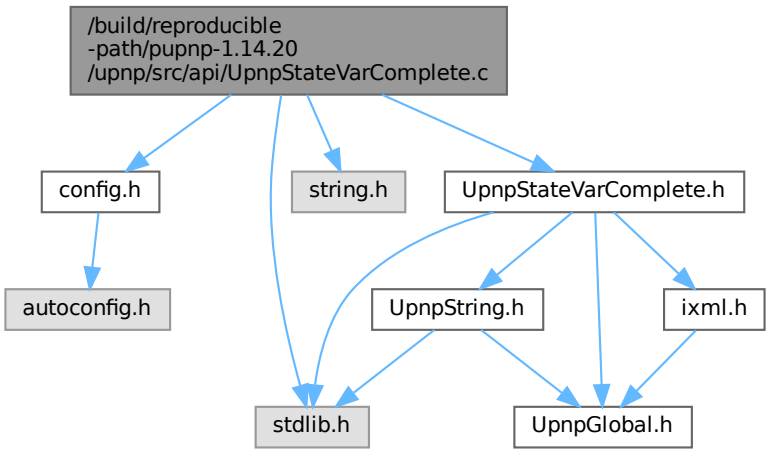
Referenced by process_request().

9.82 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarComplete.c File Reference

Source file for UpnpStateVarComplete methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpStateVarComplete.h"
```

Include dependency graph for UpnpStateVarComplete.c:



Data Structures

- struct s_UpnpStateVarComplete

Functions

- UpnpStateVarComplete * UpnpStateVarComplete_new ()
- void UpnpStateVarComplete_delete (UpnpStateVarComplete *q)
- int UpnpStateVarComplete_assign (UpnpStateVarComplete *p, const UpnpStateVarComplete *q)
- UpnpStateVarComplete * UpnpStateVarComplete_dup (const UpnpStateVarComplete *q)
- int UpnpStateVarComplete_get_ErrCode (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_ErrCode (UpnpStateVarComplete *p, int n)
- const UpnpString * UpnpStateVarComplete_get_CtrlUrl (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_CtrlUrl (UpnpStateVarComplete *p, const UpnpString *s)
- size_t UpnpStateVarComplete_get_CtrlUrl_Length (const UpnpStateVarComplete *p)
- const char * UpnpStateVarComplete_get_CtrlUrl_cstr (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_strcpy_CtrlUrl (UpnpStateVarComplete *p, const char *s)
- int UpnpStateVarComplete_strncpy_CtrlUrl (UpnpStateVarComplete *p, const char *s, size_t n)
- void UpnpStateVarComplete_clear_CtrlUrl (UpnpStateVarComplete *p)
- const UpnpString * UpnpStateVarComplete_get_StateVarName (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_StateVarName (UpnpStateVarComplete *p, const UpnpString *s)
- size_t UpnpStateVarComplete_get_StateVarName_Length (const UpnpStateVarComplete *p)
- const char * UpnpStateVarComplete_get_StateVarName_cstr (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_strcpy_StateVarName (UpnpStateVarComplete *p, const char *s)
- int UpnpStateVarComplete_strncpy_StateVarName (UpnpStateVarComplete *p, const char *s, size_t n)
- void UpnpStateVarComplete_clear_StateVarName (UpnpStateVarComplete *p)
- const DOMString UpnpStateVarComplete_get_CurrentVal (const UpnpStateVarComplete *p)
- int UpnpStateVarComplete_set_CurrentVal (UpnpStateVarComplete *p, const DOMString s)
- const char * UpnpStateVarComplete_get_CurrentVal_cstr (const UpnpStateVarComplete *p)

9.82.1 Detailed Description

Source file for UpnpStateVarComplete methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.82.2 Function Documentation

9.82.2.1 UpnpStateVarComplete_assign()

```
int UpnpStateVarComplete_assign (
            UpnpStateVarComplete * p,
            const UpnpStateVarComplete * q )
```

Assignment operator

References UpnpStateVarComplete_get_CtrlUrl(), UpnpStateVarComplete_get_CurrentVal(), UpnpStateVarComplete_get_ErrCode(), UpnpStateVarComplete_get_StateVarName(), UpnpStateVarComplete_set_CtrlUrl(), UpnpStateVarComplete_set_CurrentVal(), UpnpStateVarComplete_set_ErrCode(), and UpnpStateVarComplete_set_StateVarName().

Referenced by UpnpStateVarComplete_dup().

9.82.2.2 UpnpStateVarComplete_clear_CtrlUrl()

```
void UpnpStateVarComplete_clear_CtrlUrl (
            UpnpStateVarComplete * p )
```

UpnpStateVarComplete_clear_CtrlUrl

References UpnpString_clear().

9.82.2.3 UpnpStateVarComplete_clear_StateVarName()

```
void UpnpStateVarComplete_clear_StateVarName (
            UpnpStateVarComplete * p )
```

UpnpStateVarComplete_clear_StateVarName

References UpnpString_clear().

9.82.2.4 UpnpStateVarComplete_delete()

```
void UpnpStateVarComplete_delete (
            UpnpStateVarComplete * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by UpnpThreadDistribution().

9.82.2.5 UpnpStateVarComplete_dup()

```
UpnpStateVarComplete * UpnpStateVarComplete_dup (
            const UpnpStateVarComplete * p )
```

Copy Constructor

References UpnpStateVarComplete_assign(), and UpnpStateVarComplete_new().

9.82.2.6 UpnpStateVarComplete_get_CtrlUrl()

```
const UpnpString * UpnpStateVarComplete_get_CtrlUrl (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CtrlUrl

Referenced by SampleUtil_PrintEvent(), UpnpStateVarComplete_assign(), UpnpStateVarComplete_get_CtrlUrl_cstr(), and UpnpStateVarComplete_get_CtrlUrl_Length().

9.82.2.7 UpnpStateVarComplete_get_CtrlUrl_cstr()

```
const char * UpnpStateVarComplete_get_CtrlUrl_cstr (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CtrlUrl_cstr

References UpnpStateVarComplete_get_CtrlUrl(), and UpnpString_get_String().

9.82.2.8 UpnpStateVarComplete_get_CtrlUrl_Length()

```
size_t UpnpStateVarComplete_get_CtrlUrl_Length (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CtrlUrl_Length

References UpnpStateVarComplete_get_CtrlUrl(), and UpnpString_get_Length().

9.82.2.9 UpnpStateVarComplete_get_CurrentVal()

```
const DOMString UpnpStateVarComplete_get_CurrentVal (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CurrentVal

Referenced by SampleUtil_PrintEvent(), UpnpStateVarComplete_assign(), and UpnpStateVarComplete_get_CurrentVal_cstr().

9.82.2.10 UpnpStateVarComplete_get_CurrentVal_cstr()

```
const char * UpnpStateVarComplete_get_CurrentVal_cstr (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_CurrentVal_cstr

References UpnpStateVarComplete_get_CurrentVal().

9.82.2.11 UpnpStateVarComplete_get_ErrCode()

```
int UpnpStateVarComplete_get_ErrCode (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_ErrCode

Referenced by SampleUtil_PrintEvent(), and UpnpStateVarComplete_assign().

9.82.2.12 UpnpStateVarComplete_get_StateVarName()

```
const UpnpString * UpnpStateVarComplete_get_StateVarName (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_StateVarName

Referenced by SampleUtil_PrintEvent(), UpnpStateVarComplete_assign(), UpnpStateVarComplete_get_StateVarName_cstr(), and UpnpStateVarComplete_get_StateVarName_Length().

9.82.2.13 UpnpStateVarComplete_get_StateVarName_cstr()

```
const char * UpnpStateVarComplete_get_StateVarName_cstr (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_StateVarName_cstr

References UpnpStateVarComplete_get_StateVarName(), and UpnpString_get_String().

9.82.2.14 UpnpStateVarComplete_get_StateVarName_Length()

```
size_t UpnpStateVarComplete_get_StateVarName_Length (
            const UpnpStateVarComplete * p )
```

UpnpStateVarComplete_get_StateVarName_Length

References UpnpStateVarComplete_get_StateVarName(), and UpnpString_get_Length().

9.82.2.15 UpnpStateVarComplete_new()

```
UpnpStateVarComplete * UpnpStateVarComplete_new ( )
```

Constructor

References UpnpString_new().

Referenced by UpnpStateVarComplete_dup(), and UpnpThreadDistribution().

9.82.2.16 UpnpStateVarComplete_set_CtrlUrl()

```
int UpnpStateVarComplete_set_CtrlUrl (
            UpnpStateVarComplete * p,
            const UpnpString * s )
```

UpnpStateVarComplete_set_CtrlUrl

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarComplete_assign().

9.82.2.17 UpnpStateVarComplete_set_CurrentVal()

```
int UpnpStateVarComplete_set_CurrentVal (
            UpnpStateVarComplete * p,
            const DOMString s )
```

UpnpStateVarComplete_set_CurrentVal

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by UpnpStateVarComplete_assign(), and UpnpThreadDistribution().

9.82.2.18 UpnpStateVarComplete_set_ErrCode()

```
int UpnpStateVarComplete_set_ErrCode (
            UpnpStateVarComplete * p,
            int n )
```

UpnpStateVarComplete_set_ErrCode

Referenced by UpnpStateVarComplete_assign(), and UpnpThreadDistribution().

9.82.2.19 UpnpStateVarComplete_set_StateVarName()

```
int UpnpStateVarComplete_set_StateVarName (
            UpnpStateVarComplete * p,
            const UpnpString * s )
```

UpnpStateVarComplete_set_StateVarName

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarComplete_assign().

9.82.2.20 UpnpStateVarComplete_strcpy_CtrlUrl()

```
int UpnpStateVarComplete_strcpy_CtrlUrl (
            UpnpStateVarComplete * p,
            const char * s )
```

UpnpStateVarComplete_strcpy_CtrlUrl

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.82.2.21 UpnpStateVarComplete_strcpy_StateVarName()

```
int UpnpStateVarComplete_strcpy_StateVarName (
            UpnpStateVarComplete * p,
            const char * s )
```

UpnpStateVarComplete_strcpy_StateVarName

References UpnpString_set_String().

Referenced by UpnpThreadDistribution().

9.82.2.22 UpnpStateVarComplete_strncpy_CtrlUrl()

```
int UpnpStateVarComplete_strncpy_CtrlUrl (
            UpnpStateVarComplete * p,
            const char * s,
            size_t n )
```

UpnpStateVarComplete_strncpy_CtrlUrl

References UpnpString_set_StringN().

9.82.2.23 UpnpStateVarComplete_strncpy_StateVarName()

```
int UpnpStateVarComplete_strncpy_StateVarName (
            UpnpStateVarComplete * p,
            const char * s,
            size_t n )
```

UpnpStateVarComplete_strncpy_StateVarName

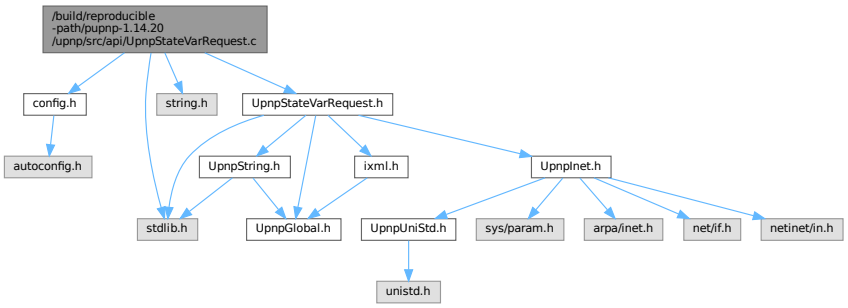
References UpnpString_set_StringN().

9.83 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarRequest.c File Reference

Source file for UpnpStateVarRequest methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpStateVarRequest.h"
```

Include dependency graph for UpnpStateVarRequest.c:



Data Structures

- struct s_UpnpStateVarRequest

Functions

- UpnpStateVarRequest * UpnpStateVarRequest_new ()
- void UpnpStateVarRequest_delete (UpnpStateVarRequest *q)
- int UpnpStateVarRequest_assign (UpnpStateVarRequest *p, const UpnpStateVarRequest *q)
- UpnpStateVarRequest * UpnpStateVarRequest_dup (const UpnpStateVarRequest *q)
- int UpnpStateVarRequest_get_ErrCode (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_ErrCode (UpnpStateVarRequest *p, int n)
- int UpnpStateVarRequest_get_Socket (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_Socket (UpnpStateVarRequest *p, int n)
- const UpnpString * UpnpStateVarRequest_get_ErrStr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_ErrStr (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_ErrStr_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_ErrStr_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_ErrStr (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_ErrStr (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_ErrStr (UpnpStateVarRequest *p)
- const UpnpString * UpnpStateVarRequest_get_DevUDN (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_DevUDN (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_DevUDN_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_DevUDN_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_DevUDN (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_DevUDN (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_DevUDN (UpnpStateVarRequest *p)
- const UpnpString * UpnpStateVarRequest_get_ServiceID (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_ServiceID (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_ServiceID_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_ServiceID_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_ServiceID (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_ServiceID (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_ServiceID (UpnpStateVarRequest *p)
- const UpnpString * UpnpStateVarRequest_get_StateVarName (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_StateVarName (UpnpStateVarRequest *p, const UpnpString *s)
- size_t UpnpStateVarRequest_get_StateVarName_Length (const UpnpStateVarRequest *p)
- const char * UpnpStateVarRequest_get_StateVarName_cstr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_strcpy_StateVarName (UpnpStateVarRequest *p, const char *s)
- int UpnpStateVarRequest_strncpy_StateVarName (UpnpStateVarRequest *p, const char *s, size_t n)
- void UpnpStateVarRequest_clear_StateVarName (UpnpStateVarRequest *p)
- const struct sockaddr_storage * UpnpStateVarRequest_get_CtrlPtIPAddr (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_CtrlPtIPAddr (UpnpStateVarRequest *p, const struct sockaddr_storage *buf)
- void UpnpStateVarRequest_clear_CtrlPtIPAddr (UpnpStateVarRequest *p)
- const DOMString UpnpStateVarRequest_get_CurrentVal (const UpnpStateVarRequest *p)
- int UpnpStateVarRequest_set_CurrentVal (UpnpStateVarRequest *p, const DOMString s)
- const char * UpnpStateVarRequest_get_CurrentVal_cstr (const UpnpStateVarRequest *p)

9.83.1 Detailed Description

Source file for UpnpStateVarRequest methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.83.2 Function Documentation

9.83.2.1 UpnpStateVarRequest_assign()

```
int UpnpStateVarRequest_assign (
            UpnpStateVarRequest * p,
            const UpnpStateVarRequest * q )
```

Assignment operator

References UpnpStateVarRequest_get_CtrlPtIPAddr(), UpnpStateVarRequest_get_CurrentVal(), UpnpStateVarRequest_get_DevUDN(), UpnpStateVarRequest_get_ErrCode(), UpnpStateVarRequest_get_ErrStr(), UpnpStateVarRequest_get_ServiceID(), UpnpStateVarRequest_get_Socket(), UpnpStateVarRequest_get_StateVarName(), UpnpStateVarRequest_set_CtrlPtIPAddr(), UpnpStateVarRequest_set_CurrentVal(), UpnpStateVarRequest_set_DevUDN(), UpnpStateVarRequest_set_ErrCode(), UpnpStateVarRequest_set_ErrStr(), UpnpStateVarRequest_set_ServiceID(), UpnpStateVarRequest_set_Socket(), and UpnpStateVarRequest_set_StateVarName().

Referenced by UpnpStateVarRequest_dup().

9.83.2.2 UpnpStateVarRequest_clear_CtrlPtIPAddr()

```
void UpnpStateVarRequest_clear_CtrlPtIPAddr (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CtrlPtIPAddr

9.83.2.3 UpnpStateVarRequest_clear_DevUDN()

```
void UpnpStateVarRequest_clear_DevUDN (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_DevUDN

References UpnpString_clear().

9.83.2.4 UpnpStateVarRequest_clear_ErrStr()

```
void UpnpStateVarRequest_clear_ErrStr (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_ErrStr

References UpnpString_clear().

9.83.2.5 UpnpStateVarRequest_clear_ServiceID()

```
void UpnpStateVarRequest_clear_ServiceID (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_ServiceID

References UpnpString_clear().

9.83.2.6 UpnpStateVarRequest_clear_StateVarName()

```
void UpnpStateVarRequest_clear_StateVarName (
            UpnpStateVarRequest * p )
```

UpnpStateVarRequest_clear_StateVarName

References UpnpString_clear().

9.83.2.7 UpnpStateVarRequest_delete()

```
void UpnpStateVarRequest_delete (
            UpnpStateVarRequest * p )
```

Destructor

References ixmlFreeDOMString(), and UpnpString_delete().

Referenced by handle_query_variable().

9.83.2.8 UpnpStateVarRequest_dup()

```
UpnpStateVarRequest * UpnpStateVarRequest_dup (
            const UpnpStateVarRequest * p )
```

Copy Constructor

References UpnpStateVarRequest_assign(), and UpnpStateVarRequest_new().

9.83.2.9 UpnpStateVarRequest_get_CtrlPtIPAddr()

```
const struct sockaddr_storage * UpnpStateVarRequest_get_CtrlPtIPAddr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CtrlPtIPAddr

Referenced by UpnpStateVarRequest_assign().

9.83.2.10 UpnpStateVarRequest_get_CurrentVal()

```
const DOMString UpnpStateVarRequest_get_CurrentVal (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CurrentVal

Referenced by handle_query_variable(), SampleUtil_PrintEvent(), UpnpStateVarRequest_assign(), and UpnpStateVarRequest_get_CurrentVal_cstr().

9.83.2.11 UpnpStateVarRequest_get_CurrentVal_cstr()

```
const char * UpnpStateVarRequest_get_CurrentVal_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_CurrentVal_cstr

References UpnpStateVarRequest_get_CurrentVal().

9.83.2.12 UpnpStateVarRequest_get_DevUDN()

```
const UpnpString * UpnpStateVarRequest_get_DevUDN (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_DevUDN

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_DevUDN_cstr(), and UpnpStateVarRequest_get_DevUDN_Length().

9.83.2.13 UpnpStateVarRequest_get_DevUDN_cstr()

```
const char * UpnpStateVarRequest_get_DevUDN_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_DevUDN_cstr

References UpnpStateVarRequest_get_DevUDN(), and UpnpString_get_String().

9.83.2.14 UpnpStateVarRequest_get_DevUDN_Length()

```
size_t UpnpStateVarRequest_get_DevUDN_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_DevUDN_Length

References UpnpStateVarRequest_get_DevUDN(), and UpnpString_get_Length().

9.83.2.15 UpnpStateVarRequest_get_ErrCode()

```
int UpnpStateVarRequest_get_ErrCode (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrCode

Referenced by handle_query_variable(), SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), and UpnpStateVarRequest_assign().

9.83.2.16 UpnpStateVarRequest_get_ErrStr()

```
const UpnpString * UpnpStateVarRequest_get_ErrStr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrStr

Referenced by handle_query_variable(), SampleUtil_PrintEvent(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_ErrStr and UpnpStateVarRequest_get_ErrStr_Length().

9.83.2.17 UpnpStateVarRequest_get_ErrStr_cstr()

```
const char * UpnpStateVarRequest_get_ErrStr_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrStr_cstr

References UpnpStateVarRequest_get_ErrStr(), and UpnpString_get_String().

Referenced by handle_query_variable().

9.83.2.18 UpnpStateVarRequest_get_ErrStr_Length()

```
size_t UpnpStateVarRequest_get_ErrStr_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ErrStr_Length

References UpnpStateVarRequest_get_ErrStr(), and UpnpString_get_Length().

9.83.2.19 UpnpStateVarRequest_get_ServiceID()

```
const UpnpString * UpnpStateVarRequest_get_ServiceID (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ServiceID

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_ServiceID_cstr(), and UpnpStateVarRequest_get_ServiceID_Length().

9.83.2.20 UpnpStateVarRequest_get_ServiceID_cstr()

```
const char * UpnpStateVarRequest_get_ServiceID_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ServiceID_cstr

References UpnpStateVarRequest_get_ServiceID(), and UpnpString_get_String().

9.83.2.21 UpnpStateVarRequest_get_ServiceID_Length()

```
size_t UpnpStateVarRequest_get_ServiceID_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_ServiceID_Length

References UpnpStateVarRequest_get_ServiceID(), and UpnpString_get_Length().

9.83.2.22 UpnpStateVarRequest_get_Socket()

```
int UpnpStateVarRequest_get_Socket (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_Socket

Referenced by UpnpStateVarRequest_assign().

9.83.2.23 UpnpStateVarRequest_get_StateVarName()

```
const UpnpString * UpnpStateVarRequest_get_StateVarName (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_StateVarName

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleGetVarRequest(), UpnpStateVarRequest_assign(), UpnpStateVarRequest_get_StateVarName_cstr(), and UpnpStateVarRequest_get_StateVarName_Length().

9.83.2.24 UpnpStateVarRequest_get_StateVarName_cstr()

```
const char * UpnpStateVarRequest_get_StateVarName_cstr (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_StateVarName_cstr

References UpnpStateVarRequest_get_StateVarName(), and UpnpString_get_String().

9.83.2.25 UpnpStateVarRequest_get_StateVarName_Length()

```
size_t UpnpStateVarRequest_get_StateVarName_Length (
            const UpnpStateVarRequest * p )
```

UpnpStateVarRequest_get_StateVarName_Length

References UpnpStateVarRequest_get_StateVarName(), and UpnpString_get_Length().

9.83.2.26 UpnpStateVarRequest_new()

```
UpnpStateVarRequest * UpnpStateVarRequest_new ( )
```

Constructor

References UpnpString_new().

Referenced by handle_query_variable(), and UpnpStateVarRequest_dup().

9.83.2.27 UpnpStateVarRequest_set_CtrlPtIPAddr()

```
int UpnpStateVarRequest_set_CtrlPtIPAddr (
            UpnpStateVarRequest * p,
            const struct sockaddr_storage * buf )
```

UpnpStateVarRequest_get_CtrlPtIPAddr

Referenced by handle_query_variable(), and UpnpStateVarRequest_assign().

9.83.2.28 UpnpStateVarRequest_set_CurrentVal()

```
int UpnpStateVarRequest_set_CurrentVal (
            UpnpStateVarRequest * p,
            const DOMString s )
```

UpnpStateVarRequest_set_CurrentVal

References DOMString, ixmlCloneDOMString(), and ixmlFreeDOMString().

Referenced by TvDeviceHandleGetVarRequest(), and UpnpStateVarRequest_assign().

9.83.2.29 UpnpStateVarRequest_set_DevUDN()

```
int UpnpStateVarRequest_set_DevUDN (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_DevUDN

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.83.2.30 UpnpStateVarRequest_set_ErrCode()

```
int UpnpStateVarRequest_set_ErrCode (
            UpnpStateVarRequest * p,
            int n )
```

UpnpStateVarRequest_set_ErrCode

Referenced by handle_query_variable(), TvDeviceHandleGetVarRequest(), and UpnpStateVarRequest_assign().

9.83.2.31 UpnpStateVarRequest_set_ErrStr()

```
int UpnpStateVarRequest_set_ErrStr (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_ErrStr

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.83.2.32 UpnpStateVarRequest_set_ServiceID()

```
int UpnpStateVarRequest_set_ServiceID (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_ServiceID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.83.2.33 UpnpStateVarRequest_set_Socket()

```
int UpnpStateVarRequest_set_Socket (
            UpnpStateVarRequest * p,
            int n )
```

UpnpStateVarRequest_set_Socket

Referenced by UpnpStateVarRequest_assign().

9.83.2.34 UpnpStateVarRequest_set_StateVarName()

```
int UpnpStateVarRequest_set_StateVarName (
            UpnpStateVarRequest * p,
            const UpnpString * s )
```

UpnpStateVarRequest_set_StateVarName

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpStateVarRequest_assign().

9.83.2.35 UpnpStateVarRequest_strcpy_DevUDN()

```
int UpnpStateVarRequest_strcpy_DevUDN (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_DevUDN

References UpnpString_set_String().

Referenced by handle_query_variable().

9.83.2.36 UpnpStateVarRequest_strcpy_ErrStr()

```
int UpnpStateVarRequest_strcpy_ErrStr (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_ErrStr

References UpnpString_set_String().

Referenced by TvDeviceHandleGetVarRequest().

9.83.2.37 UpnpStateVarRequest_strcpy_ServiceID()

```
int UpnpStateVarRequest_strcpy_ServiceID (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_ServiceID

References UpnpString_set_String().

Referenced by handle_query_variable().

9.83.2.38 UpnpStateVarRequest_strcpy_StateVarName()

```
int UpnpStateVarRequest_strcpy_StateVarName (
            UpnpStateVarRequest * p,
            const char * s )
```

UpnpStateVarRequest_strcpy_StateVarName

References UpnpString_set_String().

Referenced by handle_query_variable().

9.83.2.39 UpnpStateVarRequest_strncpy_DevUDN()

```
int UpnpStateVarRequest_strncpy_DevUDN (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_DevUDN

References UpnpString_set_StringN().

9.83.2.40 UpnpStateVarRequest_strncpy_ErrStr()

```
int UpnpStateVarRequest_strncpy_ErrStr (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_ErrStr

References UpnpString_set_StringN().

9.83.2.41 UpnpStateVarRequest_strncpy_ServiceID()

```
int UpnpStateVarRequest_strncpy_ServiceID (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_ServiceID

References UpnpString_set_StringN().

9.83.2.42 UpnpStateVarRequest_strncpy_StateVarName()

```
int UpnpStateVarRequest_strncpy_StateVarName (
            UpnpStateVarRequest * p,
            const char * s,
            size_t n )
```

UpnpStateVarRequest_strncpy_StateVarName

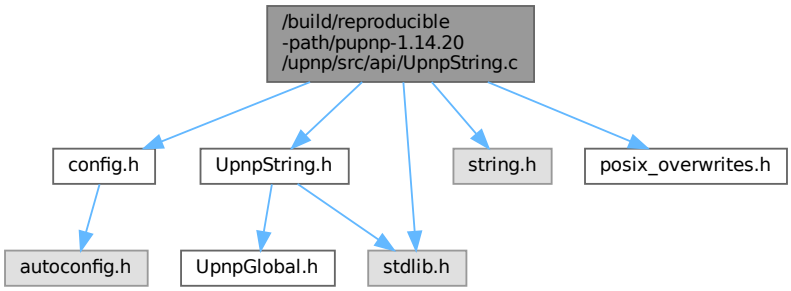
References UpnpString_set_StringN().

9.84 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpString.c File Reference

UpnpString object implementation.

```
#include "config.h"
#include "UpnpString.h"
#include <stdlib.h>
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for UpnpString.c:



Data Structures

- struct SUpnpString

  Internal implementation of the class UpnpString.

Functions

- static size_t **strnlen** (const char *s, size_t n)
- static char * **strndup** (const char *__string, size_t __n)
- UpnpString * UpnpString_new (void)

  Constructor.
- void UpnpString_delete (UpnpString *p)

  Destructor.
- UpnpString * UpnpString_dup (const UpnpString *p)

  Copy Constructor.
- void UpnpString_assign (UpnpString *p, const UpnpString *q)

  Assignment operator.
- size_t UpnpString_get_Length (const UpnpString *p)

  Returns the length of the string.
- void UpnpString_set_Length (UpnpString *p, size_t n)

  Truncates the string to the specified lenght, or does nothing if the current lenght is less than or equal to the requested length.
- const char * UpnpString_get_String (const UpnpString *p)

  Returns the pointer to char.
- int UpnpString_set_String (UpnpString *p, const char *s)

Sets the string from a pointer to char.

- int UpnpString_set_StringN (UpnpString *p, const char *s, size_t n)

  Sets the string from a pointer to char using a maximum of N chars.

- void UpnpString_clear (UpnpString *p)

  Clears the string, sets its size to zero.

- int UpnpString_cmp (UpnpString *p, UpnpString *q)

  Compares two strings for equality. Case matters.

- int UpnpString_casecmp (UpnpString *p, UpnpString *q)

  Compares two strings for equality. Case does not matter.

9.84.1 Detailed Description

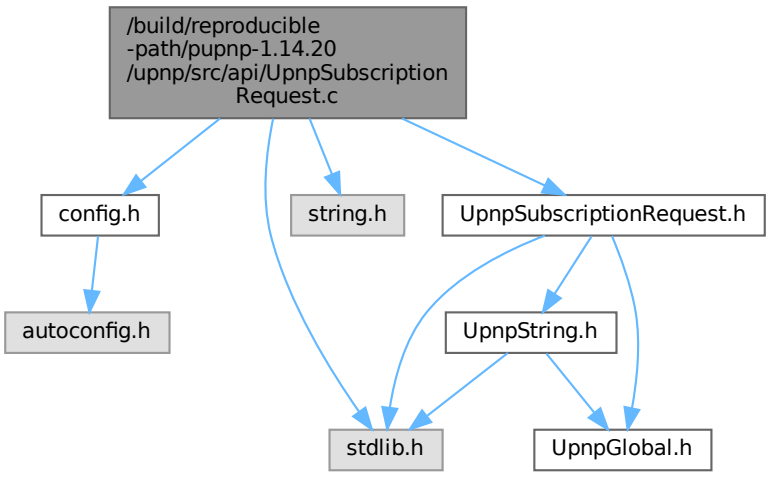
UpnpString object implementation.

9.85 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpSubscriptionRequest.c File Reference

Source file for UpnpSubscriptionRequest methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "UpnpSubscriptionRequest.h"
```

Include dependency graph for UpnpSubscriptionRequest.c:



Data Structures

- struct s_UpnpSubscriptionRequest

Functions

- UpnpSubscriptionRequest * UpnpSubscriptionRequest_new ()
- void UpnpSubscriptionRequest_delete (UpnpSubscriptionRequest *q)
- int UpnpSubscriptionRequest_assign (UpnpSubscriptionRequest *p, const UpnpSubscriptionRequest *q)
- UpnpSubscriptionRequest * UpnpSubscriptionRequest_dup (const UpnpSubscriptionRequest *q)
- const UpnpString * UpnpSubscriptionRequest_get_ServiceId (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_set_ServiceId (UpnpSubscriptionRequest *p, const UpnpString *s)
- size_t UpnpSubscriptionRequest_get_ServiceId_Length (const UpnpSubscriptionRequest *p)
- const char * UpnpSubscriptionRequest_get_ServiceId_cstr (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_strcpy_ServiceId (UpnpSubscriptionRequest *p, const char *s)
- int UpnpSubscriptionRequest_strncpy_ServiceId (UpnpSubscriptionRequest *p, const char *s, size_t n)
- void UpnpSubscriptionRequest_clear_ServiceId (UpnpSubscriptionRequest *p)
- const UpnpString * UpnpSubscriptionRequest_get_UDN (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_set_UDN (UpnpSubscriptionRequest *p, const UpnpString *s)
- size_t UpnpSubscriptionRequest_get_UDN_Length (const UpnpSubscriptionRequest *p)
- const char * UpnpSubscriptionRequest_get_UDN_cstr (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_strcpy_UDN (UpnpSubscriptionRequest *p, const char *s)
- int UpnpSubscriptionRequest_strncpy_UDN (UpnpSubscriptionRequest *p, const char *s, size_t n)
- void UpnpSubscriptionRequest_clear_UDN (UpnpSubscriptionRequest *p)
- const UpnpString * UpnpSubscriptionRequest_get_SID (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_set_SID (UpnpSubscriptionRequest *p, const UpnpString *s)
- size_t UpnpSubscriptionRequest_get_SID_Length (const UpnpSubscriptionRequest *p)
- const char * UpnpSubscriptionRequest_get_SID_cstr (const UpnpSubscriptionRequest *p)
- int UpnpSubscriptionRequest_strcpy_SID (UpnpSubscriptionRequest *p, const char *s)
- int UpnpSubscriptionRequest_strncpy_SID (UpnpSubscriptionRequest *p, const char *s, size_t n)
- void UpnpSubscriptionRequest_clear_SID (UpnpSubscriptionRequest *p)

9.85.1 Detailed Description

Source file for UpnpSubscriptionRequest methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.85.2 Function Documentation

9.85.2.1 UpnpSubscriptionRequest_assign()

```
int UpnpSubscriptionRequest_assign (
            UpnpSubscriptionRequest * p,
            const UpnpSubscriptionRequest * q )
```

Assignment operator

References UpnpSubscriptionRequest_get_ServiceId(), UpnpSubscriptionRequest_get_SID(), UpnpSubscriptionRequest_get_UDN(), UpnpSubscriptionRequest_set_ServiceId(), UpnpSubscriptionRequest_set_SID(), and UpnpSubscriptionRequest_set_UDN().

Referenced by UpnpSubscriptionRequest_dup().

9.85.2.2 UpnpSubscriptionRequest_clear_ServiceId()

```
void UpnpSubscriptionRequest_clear_ServiceId (
            UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_clear_ServiceId

References UpnpString_clear().

9.85.2.3 UpnpSubscriptionRequest_clear_SID()

```
void UpnpSubscriptionRequest_clear_SID (
            UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_clear_SID

References UpnpString_clear().

9.85.2.4 UpnpSubscriptionRequest_clear_UDN()

```
void UpnpSubscriptionRequest_clear_UDN (
            UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_clear_UDN

References UpnpString_clear().

9.85.2.5 UpnpSubscriptionRequest_delete()

```
void UpnpSubscriptionRequest_delete (
            UpnpSubscriptionRequest * p )
```

Destructor

References UpnpString_delete().

Referenced by gena_process_subscription_request().

9.85.2.6 UpnpSubscriptionRequest_dup()

```
UpnpSubscriptionRequest * UpnpSubscriptionRequest_dup (
            const UpnpSubscriptionRequest * p )
```

Copy Constructor

References UpnpSubscriptionRequest_assign(), and UpnpSubscriptionRequest_new().

9.85.2.7 UpnpSubscriptionRequest_get_ServiceId()

```
const UpnpString * UpnpSubscriptionRequest_get_ServiceId (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_ServiceId

Referenced by SampleUtil_PrintEvent(), TvDeviceHandleSubscriptionRequest(), UpnpSubscriptionRequest_assign(), UpnpSubscriptionRequest_get_ServiceId_cstr(), and UpnpSubscriptionRequest_get_ServiceId_Length().

9.85.2.8 UpnpSubscriptionRequest_get_ServiceId_cstr()

```
const char * UpnpSubscriptionRequest_get_ServiceId_cstr (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_ServiceId_cstr

References UpnpString_get_String(), and UpnpSubscriptionRequest_get_ServiceId().

9.85.2.9 UpnpSubscriptionRequest_get_ServiceId_Length()

```
size_t UpnpSubscriptionRequest_get_ServiceId_Length (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_ServiceId_Length

References UpnpString_get_Length(), and UpnpSubscriptionRequest_get_ServiceId().

9.85.2.10 UpnpSubscriptionRequest_get_SID()

```
const UpnpString * UpnpSubscriptionRequest_get_SID (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_SID

Referenced by SampleUtil_PrintEvent(), UpnpSubscriptionRequest_assign(), UpnpSubscriptionRequest_get_SID_cstr(), and UpnpSubscriptionRequest_get_SID_Length().

9.85.2.11 UpnpSubscriptionRequest_get_SID_cstr()

```
const char * UpnpSubscriptionRequest_get_SID_cstr (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_SID_cstr

References UpnpString_get_String(), and UpnpSubscriptionRequest_get_SID().

Referenced by TvDeviceHandleSubscriptionRequest().

9.85.2.12 UpnpSubscriptionRequest_get_SID_Length()

```
size_t UpnpSubscriptionRequest_get_SID_Length (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_SID_Length

References UpnpString_get_Length(), and UpnpSubscriptionRequest_get_SID().

9.85.2.13 UpnpSubscriptionRequest_get_UDN()

```
const UpnpString * UpnpSubscriptionRequest_get_UDN (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_UDN

Referenced by SampleUtil_PrintEvent(), UpnpSubscriptionRequest_assign(), UpnpSubscriptionRequest_get_UDN_cstr(), and UpnpSubscriptionRequest_get_UDN_Length().

9.85.2.14 UpnpSubscriptionRequest_get_UDN_cstr()

```
const char * UpnpSubscriptionRequest_get_UDN_cstr (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_UDN_cstr

References UpnpString_get_String(), and UpnpSubscriptionRequest_get_UDN().

Referenced by TvDeviceHandleSubscriptionRequest().

9.85.2.15 UpnpSubscriptionRequest_get_UDN_Length()

```
size_t UpnpSubscriptionRequest_get_UDN_Length (
            const UpnpSubscriptionRequest * p )
```

UpnpSubscriptionRequest_get_UDN_Length

References UpnpString_get_Length(), and UpnpSubscriptionRequest_get_UDN().

9.85.2.16 UpnpSubscriptionRequest_new()

```
UpnpSubscriptionRequest * UpnpSubscriptionRequest_new ( )
```

Constructor

References UpnpString_new().

Referenced by gena_process_subscription_request(), and UpnpSubscriptionRequest_dup().

9.85.2.17 UpnpSubscriptionRequest_set_ServiceId()

```
int UpnpSubscriptionRequest_set_ServiceId (
            UpnpSubscriptionRequest * p,
            const UpnpString * s )
```

UpnpSubscriptionRequest_set_ServiceId

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpSubscriptionRequest_assign().

9.85.2.18 UpnpSubscriptionRequest_set_SID()

```
int UpnpSubscriptionRequest_set_SID (
            UpnpSubscriptionRequest * p,
            const UpnpString * s )
```

UpnpSubscriptionRequest_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpSubscriptionRequest_assign().

9.85.2.19 UpnpSubscriptionRequest_set_UDN()

```
int UpnpSubscriptionRequest_set_UDN (
            UpnpSubscriptionRequest * p,
            const UpnpString * s )
```

UpnpSubscriptionRequest_set_UDN

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by UpnpSubscriptionRequest_assign().

9.85.2.20 UpnpSubscriptionRequest_strcpy_ServiceId()

```
int UpnpSubscriptionRequest_strcpy_ServiceId (
            UpnpSubscriptionRequest * p,
            const char * s )
```

UpnpSubscriptionRequest_strcpy_ServiceId

References UpnpString_set_String().

Referenced by gena_process_subscription_request().

9.85.2.21 UpnpSubscriptionRequest_strcpy_SID()

```
int UpnpSubscriptionRequest_strcpy_SID (
            UpnpSubscriptionRequest * p,
            const char * s )
```

UpnpSubscriptionRequest_strcpy_SID

References UpnpString_set_String().

Referenced by gena_process_subscription_request().

9.85.2.22 UpnpSubscriptionRequest_strcpy_UDN()

```
int UpnpSubscriptionRequest_strcpy_UDN (
            UpnpSubscriptionRequest * p,
            const char * s )
```

UpnpSubscriptionRequest_strcpy_UDN

References UpnpString_set_String().

Referenced by gena_process_subscription_request().

9.85.2.23 UpnpSubscriptionRequest_strncpy_ServiceId()

```
int UpnpSubscriptionRequest_strncpy_ServiceId (
            UpnpSubscriptionRequest * p,
            const char * s,
            size_t n )
```

UpnpSubscriptionRequest_strncpy_ServiceId

References UpnpString_set_StringN().

9.85.2.24 UpnpSubscriptionRequest_strncpy_SID()

```
int UpnpSubscriptionRequest_strncpy_SID (
            UpnpSubscriptionRequest * p,
            const char * s,
            size_t n )
```

UpnpSubscriptionRequest_strncpy_SID

References UpnpString_set_StringN().

9.85.2.25 UpnpSubscriptionRequest_strncpy_UDN()

```
int UpnpSubscriptionRequest_strncpy_UDN (
            UpnpSubscriptionRequest * p,
            const char * s,
            size_t n )
```

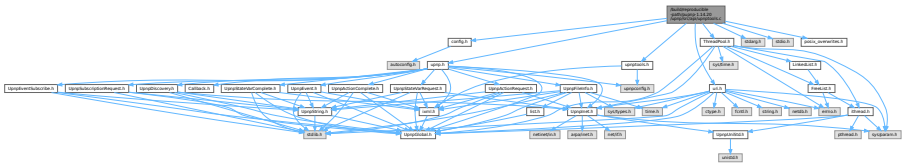
UpnpSubscriptionRequest_strncpy_UDN

References UpnpString_set_StringN().

9.86 /build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnptools.c File Reference

```
#include "config.h"
#include "upnp.h"
#include "upnptools.h"
#include "ThreadPool.h"
#include "uri.h"
#include <stdarg.h>
#include <stdio.h>
#include "posix_overwrites.h"
```

Include dependency graph for upnptools.c:



Data Structures

- struct ErrorString

  Structure to maintain a error code and string associated with the error code.

Macros

- #define HEADER_LENGTH 2000

Functions

- const char * UpnpGetErrorMessage (int rc)

  Converts an SDK error code into a string error message suitable for display. The memory returned from this function should NOT be freed.
- void UpnpSetMaxJobsTotal (int mjt)

  Sets the maximum number of jobs in the internal thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.
- int UpnpResolveURL (const char *BaseURL, const char *RelURL, char *AbsURL)

  Combines a base URL and a relative URL into a single absolute URL.
- int UpnpResolveURL2 (const char *BaseURL, const char *RelURL, char **AbsURL)

  Combines a base URL and a relative URL into a single absolute URL.
- static int addToAction (int response, IXML_Document **ActionDoc, const char *ActionName, const char *ServType, const char *ArgName, const char *ArgValue)

  Adds the argument in the action request or response.
- static IXML_Document * makeAction (int response, const char *ActionName, const char *ServType, int NumArg, const char *Arg, va_list ArgList)

  Creates the action request or response from the argument list.
- IXML_Document * UpnpMakeAction (const char *ActionName, const char *ServType, int NumArg, const char *Arg,...)

  Creates an action request packet based on its input parameters (status variable name and value pair).

- IXML_Document * UpnpMakeActionResponse (const char *ActionName, const char *ServType, int NumArg, const char *Arg,...)

  Ceates an action response packet based on its output parameters (status variable name and value pair).
- int UpnpAddToAction (IXML_Document **ActionDoc, const char *ActionName, const char *ServType, const char *ArgName, const char *ArgValue)

  Adds the argument in the action request.
- int UpnpAddToActionResponse (IXML_Document **ActionResponse, const char *ActionName, const char *ServType, const char *ArgName, const char *ArgValue)

  Creates an action response packet based on its output parameters (status variable name and value pair).
- IXML_Document * UpnpCreatePropertySet (int NumArg, const char *Arg,...)

  Creates a property set message packet.
- int UpnpAddToPropertySet (IXML_Document **PropSet, const char *ArgName, const char *ArgValue)

  Can be used when an application needs to transfer the status of many variables at once.

Variables

- struct ErrorString ErrorMessages [ ]

  Array of error structures.

9.86.1 Macro Definition Documentation

9.86.1.1 HEADER_LENGTH

```
#define HEADER_LENGTH 2000
```

Maximum action header buffer length.

9.86.2 Function Documentation

9.86.2.1 addToAction()

```
static int addToAction (
            int response,
            IXML_Document ** ActionDoc,
            const char * ActionName,
            const char * ServType,
            const char * ArgName,
            const char * ArgValue )  [static]
```

Adds the argument in the action request or response.

This function creates the action request or response if it is a first argument, otherwise it will add the argument in the document.

Returns

UPNP_E_SUCCESS if successful, otherwise the appropriate error.

Parameters

in	*response*	flag to tell if the ActionDoc is for response or request.
in,out	*ActionDoc*	Request or response document.
in	*ActionName*	Name of the action request or response.
in	*ServType*	Service type.
in	*ArgName*	Name of the argument.
in	*ArgValue*	Value of the argument.

References HEADER_LENGTH, ixmlDocument_createElement(), ixmlDocument_createTextNode(), ixmlNode_appendChild(), ixmlNode_getFirstChild(), ixmlParseBufferEx(), ErrorString::rc, UPNP_E_INVALID_DESC, UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by UpnpAddToAction(), and UpnpAddToActionResponse().

9.86.2.2 makeAction()

```
static IXML_Document * makeAction (
             int response,
             const char * ActionName,
             const char * ServType,
             int NumArg,
             const char * Arg,
             va_list ArgList )  [static]
```

Creates the action request or response from the argument list.

Returns

Action request or response document if successful, otherwise returns NULL

Parameters

in	*response*	flag to tell if the ActionDoc is for response or request.
in	*ActionName*	Name of the action request or response.
in	*ServType*	Service type.
in	*NumArg*	Number of arguments in the action request or response.
in	*Arg*	pointer to the first argument.
in	*ArgList*	Argument list.

References HEADER_LENGTH, ixmlDocument_createElement(), ixmlDocument_createTextNode(), ixmlNode_appendChild(), ixmlNode_getFirstChild(), ixmlParseBufferEx(), and ErrorString::rc.

Referenced by UpnpMakeAction(), and UpnpMakeActionResponse().

9.86.3 Variable Documentation

9.86.3.1 ErrorMessages

```
struct ErrorString ErrorMessages[]
```

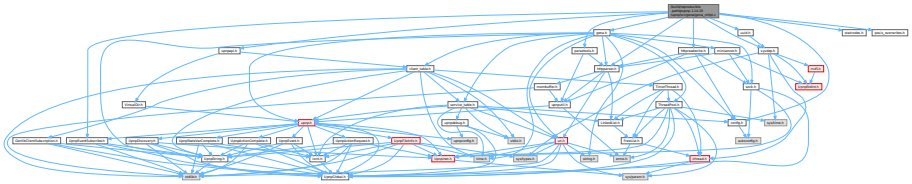
Array of error structures.

Referenced by UpnpGetErrorMessage().

9.87 /build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_ctrlpt.c File Reference

```
#include "config.h"
#include "UpnpEventSubscribe.h"
#include "gena.h"
#include "httpparser.h"
#include "httpreadwrite.h"
#include "parsetools.h"
#include "statcodes.h"
#include "sysdep.h"
#include "upnpapi.h"
#include "uuid.h"
#include "posix_overwrites.h"
```

Include dependency graph for gena_ctrlpt.c:



Data Structures

- struct job_arg

Functions

- static void free_subscribe_arg (job_arg *arg)

  Free memory associated with job's argument.
- static void GenaAutoRenewSubscription (void *input)

  This is a thread function to send the renewal just before the subscription times out.
- static int ScheduleGenaAutoRenew (int client_handle, int TimeOut, GenlibClientSubscription *sub)

  Schedules a job to renew the subscription just before time out.
- static int gena_unsubscribe (const UpnpString *url, const UpnpString *sid, http_parser_t *response)

  Sends the UNSUBCRIBE gena request and recieves the response from the device and returns it as a parameter.
- static int gena_subscribe (const UpnpString *url, int *timeout, const UpnpString *renewal_sid, UpnpString *sid)

  Subscribes or renew subscription.
- int genaUnregisterClient (UpnpClient_Handle client_handle)

  Unsubcribes all the outstanding subscriptions and cleans the subscription list.
- int genaUnSubscribe (UpnpClient_Handle client_handle, const UpnpString *in_sid)

  Unsubscribes a SID.

- int genaSubscribe (UpnpClient_Handle client_handle, const UpnpString *PublisherURL, int *TimeOut, UpnpString *out_sid)

  This function subscribes to a PublisherURL (also mentioned as EventURL in some places).
- int genaRenewSubscription (UpnpClient_Handle client_handle, const UpnpString *in_sid, int *TimeOut)

  Renews a SID.
- void gena_process_notification_event (SOCKINFO *info, http_message_t *event)

  This function processes NOTIFY events that are sent by devices.

Variables

- ithread_mutex_t GlobalClientSubscribeMutex

9.87.1 Function Documentation

9.87.1.1 free_subscribe_arg()

```
static void free_subscribe_arg (
            job_arg * arg )  [static]
```

Free memory associated with job's argument.

References UpnpEventSubscribe_delete().

Referenced by GenaAutoRenewSubscription(), and ScheduleGenaAutoRenew().

9.87.1.2 gena_process_notification_event()

```
void gena_process_notification_event (
            SOCKINFO * info,
            http_message_t * event )
```

This function processes NOTIFY events that are sent by devices.

Parameters: IN SOCKINFO *info: Socket structure containing the device socket information IN http_message_t *event: The http message contains the GENA notification

Note

called by genacallback()

Parameters

in	*info*	Socket info of the device.
in	*event*	The http message contains the GENA notification.

References memptr::buf, Handle_Info::Callback, Handle_Info::ClientSubList, Handle_Info::Cookie, http_message_t::entity, GenlibClientSubscription_get_SID(), GetClientHandleInfo(), GetHandleInfo(), has_xml_content_type(), ixmlDocument_free(), ixmlParseBufferEx(), memptr::length, membuffer::length, http_message_t::msg, PARSE_OK, SubscribeLock,

SubscribeUnlock, UPNP_EVENT_RECEIVED, UpnpEvent_delete(), UpnpEvent_new(), UpnpEvent_set_ChangedVariables(), UpnpEvent_set_EventKey(), and UpnpEvent_set_SID().

Referenced by genaCallback().

9.87.1.3 gena_subscribe()

```
static int gena_subscribe (
            const UpnpString * url,
            int * timeout,
            const UpnpString * renewal_sid,
            UpnpString * sid )  [static]
```

Subscribes or renew subscription.

Returns

0 if successful, otherwise returns the appropriate error code.

Parameters

in	*url*	URL of service to subscribe.
in, out	*timeout*	Subscription time desired (in secs).
in	*renewal_sid*	for renewal, this contains a currently held subscription SID. For first time subscription, this must be NULL.
out	*sid*	SID returned by the subscription or renew msg.

References memptr::buf, membuffer::buf, gIF_IPV4, gIF_IPV6, gIF_IPV6_ULA_GUA, http_FixStrUrl(), http_MakeMessage(), memptr::length, membuffer::length, LOCAL_PORT_V4, LOCAL_PORT_V6, LOCAL_PORT_V6_ULA_GUA, PARSE_OK, membuffer::size_inc, http_message_t::status_code, UPNP_E_BAD_RESPONSE, UPNP_E_OUTOF_MEMORY, UPNP_E_SUBSCRIBE_UNACCEPTED, UPNP_E_SUCCESS, UpnpString_clear(), UpnpString_get_Length(), UpnpString_get_String(), and UpnpString_set_StringN().

Referenced by genaRenewSubscription(), and genaSubscribe().

9.87.1.4 gena_unsubscribe()

```
static int gena_unsubscribe (
            const UpnpString * url,
            const UpnpString * sid,
            http_parser_t * response )  [static]
```

Sends the UNSUBCRIBE gena request and recieves the response from the device and returns it as a parameter.

Returns

0 if successful, otherwise returns the appropriate error code.

Parameters

in	*url*	Event URL of the service.
in	*sid*	The subcription ID.
out	*response*	The UNSUBCRIBE response from the device.

References membuffer::buf, http_FixStrUrl(), http_MakeMessage(), membuffer::length, membuffer::size_inc, http_message_t::status_code, UPNP_E_UNSUBSCRIBE_UNACCEPTED, UpnpString_get_Length(), and UpnpString_get_String().

Referenced by genaUnregisterClient(), and genaUnSubscribe().

9.87.1.5 GenaAutoRenewSubscription()

```
static void GenaAutoRenewSubscription (
            void * input )  [static]
```

This is a thread function to send the renewal just before the subscription times out.

Parameters

in	*input*	Thread data(job_arg *) needed to send the renewal.

References Handle_Info::Callback, Handle_Info::Cookie, free_subscribe_arg(), genaRenewSubscription(), GetHandleInfo(), UPNP_E_SUCCESS, UPNP_EVENT_AUTORENEWAL_FAILED, UPNP_EVENT_SUBSCRIPTION_EXPIRED, UpnpEventSubscribe_get_SID(), UpnpEventSubscribe_get_TimeOut(), UpnpEventSubscribe_set_ErrCode(), UpnpEventSubscribe_set_TimeOut(), and UpnpPrintf().

Referenced by ScheduleGenaAutoRenew().

9.87.1.6 genaRenewSubscription()

```
int genaRenewSubscription (
            UpnpClient_Handle client_handle,
            const UpnpString * in_sid,
            int * TimeOut )
```

Renews a SID.

It first validates the SID and client_handle and copies the subscription. It sends RENEW (modified SUBSCRIBE) http request to service and processes the response.

Returns

UPNP_E_SUCCESS if service response is OK, otherwise the appropriate error code.

Parameters

in	*client_handle*	Client handle.
in	*in_sid*	Subscription ID.
in,out	*TimeOut*	requested Duration, if -1, then "infinite". In the OUT case: actual Duration granted by Service, -1 for infinite.

References Handle_Info::ClientSubList, gena_subscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_delete(), GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_get_RenewEventId(), GenlibClientSubscription_get_SID(), GenlibClientSubscription_new(), GenlibClientSubscription_set_ActualSID(),

GenlibClientSubscription_set_RenewEventId(), GetHandleInfo(), gTimerThread, ScheduleGenaAutoRenew(), TimerThreadRemove(), UPNP_E_SUCCESS, UpnpPrintf(), UpnpString_delete(), and UpnpString_new().

Referenced by GenaAutoRenewSubscription(), UpnpRenewSubscription(), and UpnpThreadDistribution().

9.87.1.7 genaSubscribe()

```
int genaSubscribe (
            UpnpClient_Handle client_handle,
            const UpnpString * PublisherURL,
            int * TimeOut,
            UpnpString * out_sid )
```

This function subscribes to a PublisherURL (also mentioned as EventURL in some places).

It sends SUBSCRIBE http request to service processes request. Finally adds a Subscription to the clients subscription list, if service responds with OK.

Returns

UPNP_E_SUCCESS if service response is OK, otherwise returns the appropriate error code

Parameters

in	*client_handle*	The client handle.
in	*PublisherURL*	Of the form: " http://134.134.156.80:4000/RedBulb/Event
in,out	*TimeOut*	requested Duration: • if -1, then "infinite". • in the OUT case: actual Duration granted by Service, -1 for infinite.
out	*out_sid*	sid of subscription, memory passed in by caller.

References Handle_Info::ClientSubList, gena_subscribe(), GenlibClientSubscription_delete(), GenlibClientSubscription_new(), GenlibClientSubscription_set_ActualSID(), GenlibClientSubscription_set_EventURL(), GenlibClientSubscription_set_Next(), GenlibClientSubscription_set_RenewEventId(), GenlibClientSubscription_set_SID(), GetHandleInfo(), ScheduleGenaAutoRenew(), SubscribeLock, SubscribeUnlock, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpString_assign(), UpnpString_clear(), UpnpString_delete(), UpnpString_new(), and UpnpString_set_String().

Referenced by UpnpSubscribe(), and UpnpThreadDistribution().

9.87.1.8 genaUnregisterClient()

```
int genaUnregisterClient (
            UpnpClient_Handle client_handle )
```

Unsubcribes all the outstanding subscriptions and cleans the subscription list.

This function is called when control point unregisters.

Returns

UPNP_E_SUCCESS if successful, otherwise returns the appropriate error code.

Parameters

in	*client_handle*	Handle containing all the control point related information.

References Handle_Info::ClientSubList, gena_unsubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_delete(), GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_get_SID(), GenlibClientSubscription_new(), GetHandleInfo(), and UPNP_E_SUCCESS.

Referenced by UpnpUnRegisterClient().

9.87.1.9 genaUnSubscribe()

```
int genaUnSubscribe (
            UpnpClient_Handle client_handle,
            const UpnpString * in_sid )
```

Unsubscribes a SID.

It first validates the SID and client_handle,copies the subscription, sends UNSUBSCRIBE http request to service processes request and finally removes the subscription.

Returns

UPNP_E_SUCCESS if service response is OK, otherwise returns the appropriate error code.

Parameters

in	*client_handle*	UPnP client handle.
in	*in_sid*	The subscription ID.

References Handle_Info::ClientSubList, gena_unsubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_delete(), GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_new(), and GetHandleInfo().

Referenced by UpnpThreadDistribution(), and UpnpUnSubscribe().

9.87.1.10 ScheduleGenaAutoRenew()

```
static int ScheduleGenaAutoRenew (
            int client_handle,
            int TimeOut,
            GenlibClientSubscription * sub )  [static]
```

Schedules a job to renew the subscription just before time out.

Returns

GENA_E_SUCCESS if successful, otherwise returns the appropriate error code.

Parameters

in	*client_handle*	Handle that also contains the subscription list.
in	*TimeOut*	The time out value of the subscription.
in	*sub*	Subscription being renewed.

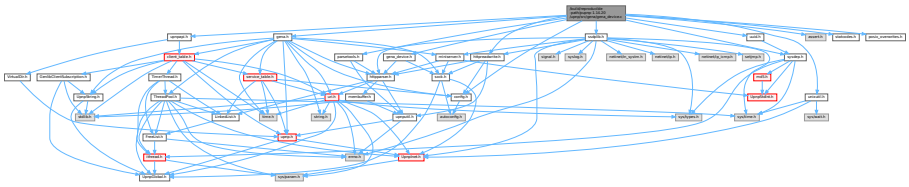
References free_subscribe_arg(), GenaAutoRenewSubscription(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_get_SID(), GenlibClientSubscription_set_RenewEventId(), gTimerThread, REL_SEC, TimerThreadSchedule(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpEventSubscribe_new(), UpnpEventSubscribe_set_ErrCode(), UpnpEventSubscribe_set_PublisherUrl(), UpnpEventSubscribe_set_SID(), and UpnpEventSubscribe_set_TimeOut().

Referenced by genaRenewSubscription(), and genaSubscribe().

9.88 /build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_device.c File Reference

```
#include "config.h"
#include "gena_device.h"
#include <assert.h>
#include "gena.h"
#include "httpreadwrite.h"
#include "parsetools.h"
#include "ssdplib.h"
#include "statcodes.h"
#include "sysdep.h"
#include "unixutil.h"
#include "upnpapi.h"
#include "uuid.h"
#include "posix_overwrites.h"
```

Include dependency graph for gena_device.c:



Macros

- #define **STALE_JOBID** (INVALID_JOB_ID - 1)

Functions

- int genaUnregisterDevice (UpnpDevice_Handle device_handle)

  Unregisters a device.
- static int GeneratePropertySet (char **names, char **values, int count, DOMString *out)

  Generates XML property set for notifications.

- static void free_notify_struct (notify_thread_struct *input)

  Frees memory used in notify_threads if the reference count is 0, otherwise decrements the refrence count.
- static UPNP_INLINE int notify_send_and_recv (uri_type *destination_url, membuffer *mid_msg, char *propertySet, http_parser_t *response)

  Sends the notify message and returns a reply.
- static int genaNotify (char *headers, char *propertySet, subscription *sub)

  Function to Notify a particular subscription of a particular event.
- static void genaNotifyThread (void *input)

  Thread job to Notify a control point.
- static char * AllocGenaHeaders (const DOMString propertySet)

  Allocates the GENA header.
- void **freeSubscriptionQueuedEvents** (subscription *sub)
- static int **genaInitNotifyCommon** (UpnpDevice_Handle device_handle, char *UDN, char *servId, DOMString propertySet, const Upnp_SID sid)
- int genaInitNotify (UpnpDevice_Handle device_handle, char *UDN, char *servId, char **VarNames, char **VarValues, int var_count, const Upnp_SID sid)

  Sends the intial state table dump to newly subscribed control point.
- int genaInitNotifyExt (UpnpDevice_Handle device_handle, char *UDN, char *servId, IXML_Document *PropSet, const Upnp_SID sid)

  Similar to the genaInitNofity. The only difference is that it takes the xml document for the state table and sends the intial state table dump to newly subscribed control point.
- static void **maybeDiscardEvents** (LinkedList *listp)
- static int **genaNotifyAllCommon** (UpnpDevice_Handle device_handle, char *UDN, char *servId, DOMString propertySet)
- int genaNotifyAllExt (UpnpDevice_Handle device_handle, char *UDN, char *servId, IXML_Document *PropSet)

  Sends a notification to all the subscribed control points.
- int genaNotifyAll (UpnpDevice_Handle device_handle, char *UDN, char *servId, char **VarNames, char **VarValues, int var_count)

  Sends a notification to all the subscribed control points.
- static int respond_ok (SOCKINFO *info, int time_out, subscription *sub, http_message_t *request)

  Returns OK message in the case of a subscription request.
- static int create_url_list (memptr *url_list, URL_list *out)

  Function to parse the Callback header value in subscription requests.
- int gena_validate_delivery_urls (SOCKINFO *info, URL_list *url_list)

  Validate that the URLs passed by the user are on the same network segment than the device.
- void gena_process_subscription_request (SOCKINFO *info, http_message_t *request)

  Handles a subscription request from a ctrl point. The socket is not closed on return.
- void gena_process_subscription_renewal_request (SOCKINFO *info, http_message_t *request)

  Handles a subscription renewal request from a ctrl point. The connection is not destroyed on return.
- void gena_process_unsubscribe_request (SOCKINFO *info, http_message_t *request)

  Handles a subscription cancellation request from a ctrl point. The connection is not destroyed on return.

9.88.1 Function Documentation

9.88.1.1 AllocGenaHeaders()

```
static char * AllocGenaHeaders (
            const DOMString propertySet )  [static]
```

Allocates the GENA header.

Note

The header must be destroyed after with a call to free(), otherwise there will be a memory leak.

Returns

The constructed header.

Parameters

in	*propertySet*	The property set string.

References UpnpPrintf().

9.88.1.2 create_url_list()

```
static int create_url_list (
             memptr * url_list,
             URL_list * out )  [static]
```

Function to parse the Callback header value in subscription requests.

Takes in a buffer containing URLS delimited by '<' and '>'. The entire buffer is copied into dynamic memory and stored in the URL_list. Pointers to the individual urls within this buffer are allocated and stored in the URL_list. Only URLs with network addresses are considered (i.e. host:port or domain name).

Returns

The number of URLs parsed if successful, otherwise UPNP_E_OUTOF_MEMORY.

Parameters

in	*url_list*	.
out	*out*	.

References memptr::buf, memptr::length, parse_uri(), HOSTPORT::text, UPNP_E_OUTOF_MEMORY, and URL_LIST::URLs.

Referenced by gena_process_subscription_request().

9.88.1.3 free_notify_struct()

```
static void free_notify_struct (
             notify_thread_struct * input )  [static]
```

Frees memory used in notify_threads if the reference count is 0, otherwise decrements the refrence count.

Parameters

in	*input*	Notify structure.

References ixmlFreeDOMString().

Referenced by genaNotifyThread().

9.88.1.4 gena_process_subscription_renewal_request()

```
void gena_process_subscription_renewal_request (
            SOCKINFO * info,
            http_message_t * request )
```

Handles a subscription renewal request from a ctrl point. The connection is not destroyed on return.

Parameters

in	*info*	Socket info of the device.
in	*request*	Subscription renewal request from the control point.

References memptr::buf, membuffer::buf, device_handle, SOCKINFO::foreign_sockaddr, GetDeviceHandleInfoForPath(), memptr::length, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, PARSE_OK, respond_ok(), UPNP_E_SUCCESS, UpnpPrintf(), and http_message_t::uri.

Referenced by genaCallback().

9.88.1.5 gena_process_subscription_request()

```
void gena_process_subscription_request (
            SOCKINFO * info,
            http_message_t * request )
```

Handles a subscription request from a ctrl point. The socket is not closed on return.

Parameters

in	*info*	Socket info of the device.
in	*request*	Subscription request from the control point.

References memptr::buf, Handle_Info::Callback, Handle_Info::Cookie, create_url_list(), device_handle, SOCKINFO::foreign_sockaddr, gena_validate_delivery_urls(), GetDeviceHandleInfoForPath(), memptr::length, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, PARSE_OK, respond_ok(), UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UPNP_EVENT_SUBSCRIPTION_REQUEST, UpnpPrintf(), UpnpSubscriptionRequest_delete(), UpnpSubscriptionRequest_new(), UpnpSubscriptionRequest_strcpy_ServiceId(), UpnpSubscriptionRequest_strcpy_SID(), UpnpSubscriptionRequest_strcpy_UDN(), http_message_t::uri, and URL_LIST::URLs.

Referenced by genaCallback().

9.88.1.6 gena_process_unsubscribe_request()

```
void gena_process_unsubscribe_request (
            SOCKINFO * info,
            http_message_t * request )
```

Handles a subscription cancellation request from a ctrl point. The connection is not destroyed on return.

Parameters

in	*info*	Socket info of the device.
in	*request*	UNSUBSCRIBE request from the control point.

References memptr::buf, membuffer::buf, device_handle, SOCKINFO::foreign_sockaddr, GetDeviceHandleInfoForPath(), memptr::length, and http_message_t::uri.

Referenced by genaCallback().

9.88.1.7 gena_validate_delivery_urls()

```
int gena_validate_delivery_urls (
            SOCKINFO * info,
            URL_list * url_list )
```

Validate that the URLs passed by the user are on the same network segment than the device.

Note: This is a fix for CallStanger a.k.a. CVE-2020-12695

Returns

0 if all URLs are on the same segment or -1 otherwise.

Parameters

in	*info*	.
in	*url_list*	.

References SOCKINFO::foreign_sockaddr, gIF_IPV4, gIF_IPV4_NETMASK, gIF_IPV6, gIF_IPV6_PREFIX_LENGTH, gIF_IPV6_ULA_GUA, gIF_IPV6_ULA_GUA_PREFIX_LENGTH, and UpnpPrintf().

Referenced by gena_process_subscription_request().

9.88.1.8 genaInitNotify()

```
int genaInitNotify (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            char ** VarNames,
            char ** VarValues,
            int var_count,
            const Upnp_SID sid )
```

Sends the intial state table dump to newly subscribed control point.

Returns

GENA_E_SUCCESS if successful, otherwise the appropriate error code.

Note

No other event will be sent to this control point before the intial state table dump.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*VarNames*	Array of variable names.
in	*VarValues*	Array of variable values.
in	*var_count*	Array size.
in	*sid*	Subscription ID.

References device_handle, DOMString, GeneratePropertySet(), and UpnpPrintf().

Referenced by UpnpAcceptSubscription().

9.88.1.9 genaInitNotifyExt()

```
int genaInitNotifyExt (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            IXML_Document * PropSet,
            const Upnp_SID sid )
```

Similar to the genaInitNofity. The only difference is that it takes the xml document for the state table and sends the intial state table dump to newly subscribed control point.

Returns

GENA_E_SUCCESS if successful, otherwise the appropriate error code.

Note

No other event will be sent to this control point before the intial state table dump.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*PropSet*	Document of the state table.
in	*sid*	subscription ID.

References device_handle, DOMString, ixmlPrintNode(), UPNP_E_INVALID_PARAM, and UpnpPrintf().

Referenced by UpnpAcceptSubscriptionExt().

9.88.1.10 genaNotify()

```
static int genaNotify (
            char * headers,
```

```
            char * propertySet,
            subscription * sub )  [static]
```

Function to Notify a particular subscription of a particular event.

In general the service should NOT be blocked around this call (this may cause deadlock with a client).

NOTIFY http request is sent and the reply is processed.

Returns

GENA_SUCCESS if the event was delivered, otherwise returns the appropriate error code.

Parameters

in	*headers*	Null terminated, includes all headers (including \r\n) except SID and SEQ.
in	*propertySet*	The evented XML.
in	*sub*	subscription to be Notified, assumes this is valid for life of function.

References http_MakeMessage(), notify_send_and_recv(), http_message_t::status_code, UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by genaNotifyThread().

9.88.1.11 genaNotifyAll()

```
int genaNotifyAll (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            char ** VarNames,
            char ** VarValues,
            int var_count )
```

Sends a notification to all the subscribed control points.

Returns

int

Note

This function is similar to the genaNotifyAllExt. The only difference is it takes event variable array instead of xml document.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*VarNames*	Array of varible names.
in	*VarValues*	Array of variable values.
in	*var_count*	Number of variables.

References device_handle, DOMString, GeneratePropertySet(), and UpnpPrintf().

Referenced by UpnpNotify().

9.88.1.12 genaNotifyAllExt()

```
int genaNotifyAllExt (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            IXML_Document * PropSet )
```

Sends a notification to all the subscribed control points.

Returns

int

Note

This function is similar to the genaNotifyAll. the only difference is it takes the document instead of event variable array.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*PropSet*	XML document Event varible property set.

References device_handle, DOMString, ixmlPrintNode(), UPNP_E_INVALID_PARAM, and UpnpPrintf().

Referenced by UpnpNotifyExt().

9.88.1.13 genaNotifyThread()

```
static void genaNotifyThread (
            void * input )  [static]
```

Thread job to Notify a control point.

It validates the subscription and copies the subscription. Also make sure that events are sent in order.

Note

calls the genaNotify to do the actual work.

Parameters

in	*input*	notify thread structure containing all the headers and property set info.

References free_notify_struct(), genaNotify(), GetHandleInfo(), gSendThreadPool, Handle_Info::ServiceTable, and ThreadPoolAdd().

9.88.1.14 genaUnregisterDevice()

```
int genaUnregisterDevice (
            UpnpDevice_Handle device_handle )
```

Unregisters a device.

Cleans the service table of the device.

Returns

UPNP_E_SUCCESS on success, GENA_E_BAD_HANDLE on failure.

Parameters

in	*device_handle*	Device handle.

References device_handle, GetHandleInfo(), Handle_Info::ServiceTable, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by UpnpUnRegisterRootDeviceLowPower().

9.88.1.15 GeneratePropertySet()

```
static int GeneratePropertySet (
            char ** names,
            char ** values,
            int count,
            DOMString * out )  [static]
```

Generates XML property set for notifications.

Returns

UPNP_E_SUCCESS if successful else returns GENA_E_BAD_HANDLE.

Note

The XML_VERSION comment is NOT sent due to interoperability issues with other UPnP vendors.

Parameters

in	*names*	Array of variable names (go in the event notify).
in	*values*	Array of variable values (go in the event notify).
in	*count*	number of variables.
out	*out*	PropertySet node in the string format.

References ixmlCloneDOMString(), and UPNP_E_OUTOF_MEMORY.

Referenced by genaInitNotify(), and genaNotifyAll().

9.88.1.16 notify_send_and_recv()

```
static UPNP_INLINE int notify_send_and_recv (
            uri_type * destination_url,
            membuffer * mid_msg,
            char * propertySet,
            http_parser_t * response )  [static]
```

Sends the notify message and returns a reply.

Returns

on success returns UPNP_E_SUCCESS, otherwise returns a UPNP error.

Note

called by genaNotify

Parameters

in	*destination_url*	subscription callback URL (URL of the control point).
in	*mid_msg*	Common HTTP headers.
in	*propertySet*	The evented XML.
out	*response*	The response from the control point.

References membuffer::buf, http_Connect(), http_MakeMessage(), http_RecvMessage(), http_SendMessage(), membuffer::length, sock_destroy(), sock_init(), HOSTPORT::text, UPNP_E_OUTOF_MEMORY, UPNP_E_SOCKET_CONNECT, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by genaNotify().

9.88.1.17 respond_ok()

```
static int respond_ok (
            SOCKINFO * info,
            int time_out,
            subscription * sub,
            http_message_t * request )  [static]
```

Returns OK message in the case of a subscription request.

Returns

UPNP_E_SUCCESS if successful, otherwise the appropriate error code.

Parameters

in	*info*	Socket connection of request.
in	*time_out*	Accepted duration.
in	*sub*	Accepted subscription.
in	*request*	Http request.

References membuffer::buf, http_MakeMessage(), http_SendMessage(), membuffer::length, membuffer::size_inc, and UPNP_E_OUTOF_MEMORY.

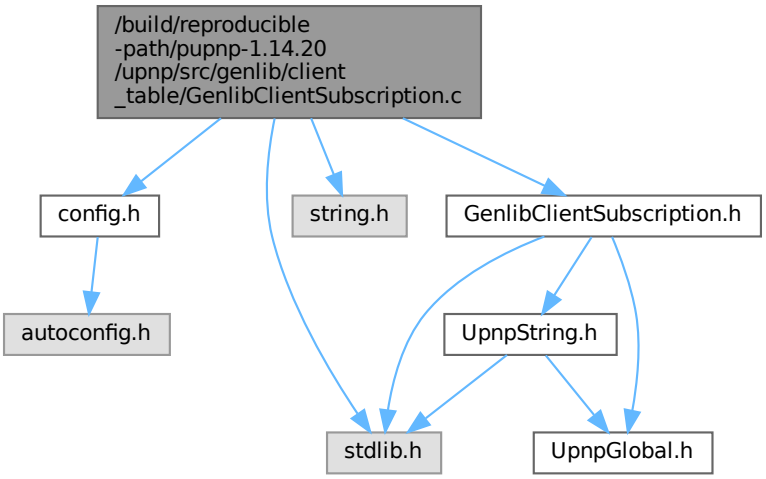
Referenced by gena_process_subscription_renewal_request(), and gena_process_subscription_request().

9.89 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/client_table/GenlibClientSubscription.c File Reference

Source file for GenlibClientSubscription methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
#include "GenlibClientSubscription.h"
```

Include dependency graph for GenlibClientSubscription.c:



Data Structures

- struct s_GenlibClientSubscription

Functions

- GenlibClientSubscription * GenlibClientSubscription_new ()
- void GenlibClientSubscription_delete (GenlibClientSubscription *q)
- int GenlibClientSubscription_assign (GenlibClientSubscription *p, const GenlibClientSubscription *q)
- GenlibClientSubscription * GenlibClientSubscription_dup (const GenlibClientSubscription *q)
- int GenlibClientSubscription_get_RenewEventId (const GenlibClientSubscription *p)
- int GenlibClientSubscription_set_RenewEventId (GenlibClientSubscription *p, int n)
- const UpnpString * GenlibClientSubscription_get_SID (const GenlibClientSubscription *p)
- int GenlibClientSubscription_set_SID (GenlibClientSubscription *p, const UpnpString *s)
- size_t GenlibClientSubscription_get_SID_Length (const GenlibClientSubscription *p)
- const char * GenlibClientSubscription_get_SID_cstr (const GenlibClientSubscription *p)
- int GenlibClientSubscription_strcpy_SID (GenlibClientSubscription *p, const char *s)
- int GenlibClientSubscription_strncpy_SID (GenlibClientSubscription *p, const char *s, size_t n)
- void GenlibClientSubscription_clear_SID (GenlibClientSubscription *p)
- const UpnpString * GenlibClientSubscription_get_ActualSID (const GenlibClientSubscription *p)
- int GenlibClientSubscription_set_ActualSID (GenlibClientSubscription *p, const UpnpString *s)
- size_t GenlibClientSubscription_get_ActualSID_Length (const GenlibClientSubscription *p)
- const char * GenlibClientSubscription_get_ActualSID_cstr (const GenlibClientSubscription *p)
- int GenlibClientSubscription_strcpy_ActualSID (GenlibClientSubscription *p, const char *s)
- int GenlibClientSubscription_strncpy_ActualSID (GenlibClientSubscription *p, const char *s, size_t n)
- void GenlibClientSubscription_clear_ActualSID (GenlibClientSubscription *p)
- const UpnpString * GenlibClientSubscription_get_EventURL (const GenlibClientSubscription *p)
- int GenlibClientSubscription_set_EventURL (GenlibClientSubscription *p, const UpnpString *s)
- size_t GenlibClientSubscription_get_EventURL_Length (const GenlibClientSubscription *p)
- const char * GenlibClientSubscription_get_EventURL_cstr (const GenlibClientSubscription *p)
- int GenlibClientSubscription_strcpy_EventURL (GenlibClientSubscription *p, const char *s)
- int GenlibClientSubscription_strncpy_EventURL (GenlibClientSubscription *p, const char *s, size_t n)
- void GenlibClientSubscription_clear_EventURL (GenlibClientSubscription *p)
- GenlibClientSubscription * GenlibClientSubscription_get_Next (const GenlibClientSubscription *p)
- int GenlibClientSubscription_set_Next (GenlibClientSubscription *p, GenlibClientSubscription *n)

9.89.1 Detailed Description

Source file for GenlibClientSubscription methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.89.2 Function Documentation

9.89.2.1 GenlibClientSubscription_assign()

```
int GenlibClientSubscription_assign (
            GenlibClientSubscription * p,
            const GenlibClientSubscription * q )
```

Assignment operator

References GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_get_Nex
GenlibClientSubscription_get_RenewEventId(), GenlibClientSubscription_get_SID(), GenlibClientSubscription_set_ActualSID(),
GenlibClientSubscription_set_EventURL(), GenlibClientSubscription_set_Next(), GenlibClientSubscription_set_RenewEventId(),
and GenlibClientSubscription_set_SID().

Referenced by genaRenewSubscription(), genaUnregisterClient(), genaUnSubscribe(), and GenlibClientSubscription_dup().

9.89.2.2 GenlibClientSubscription_clear_ActualSID()

```
void GenlibClientSubscription_clear_ActualSID (
            GenlibClientSubscription * p )
```

GenlibClientSubscription_clear_ActualSID

References UpnpString_clear().

9.89.2.3 GenlibClientSubscription_clear_EventURL()

```
void GenlibClientSubscription_clear_EventURL (
            GenlibClientSubscription * p )
```

GenlibClientSubscription_clear_EventURL

References UpnpString_clear().

9.89.2.4 GenlibClientSubscription_clear_SID()

```
void GenlibClientSubscription_clear_SID (
            GenlibClientSubscription * p )
```

GenlibClientSubscription_clear_SID

References UpnpString_clear().

9.89.2.5 GenlibClientSubscription_delete()

```
void GenlibClientSubscription_delete (
            GenlibClientSubscription * p )
```

Destructor

References UpnpString_delete().

Referenced by freeClientSubList(), genaRenewSubscription(), genaSubscribe(), genaUnregisterClient(), and genaUnSubscribe().

9.89.2.6 GenlibClientSubscription_dup()

```
GenlibClientSubscription * GenlibClientSubscription_dup (
            const GenlibClientSubscription * p )
```

Copy Constructor

References GenlibClientSubscription_assign(), and GenlibClientSubscription_new().

9.89.2.7 GenlibClientSubscription_get_ActualSID()

```
const UpnpString * GenlibClientSubscription_get_ActualSID (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_ActualSID

Referenced by genaRenewSubscription(), genaUnregisterClient(), genaUnSubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_get_ActualSID_cstr(), and GenlibClientSubscription_get_ActualSID_Length().

9.89.2.8 GenlibClientSubscription_get_ActualSID_cstr()

```
const char * GenlibClientSubscription_get_ActualSID_cstr (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_ActualSID_cstr

References GenlibClientSubscription_get_ActualSID(), and UpnpString_get_String().

Referenced by GetClientSubActualSID().

9.89.2.9 GenlibClientSubscription_get_ActualSID_Length()

```
size_t GenlibClientSubscription_get_ActualSID_Length (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_ActualSID_Length

References GenlibClientSubscription_get_ActualSID(), and UpnpString_get_Length().

9.89.2.10 GenlibClientSubscription_get_EventURL()

```
const UpnpString * GenlibClientSubscription_get_EventURL (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_EventURL

Referenced by genaRenewSubscription(), genaUnregisterClient(), genaUnSubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_get_EventURL_cstr(), GenlibClientSubscription_get_EventURL_Length(), and ScheduleGenaAutoRenew().

9.89.2.11 GenlibClientSubscription_get_EventURL_cstr()

```
const char * GenlibClientSubscription_get_EventURL_cstr (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_EventURL_cstr

References GenlibClientSubscription_get_EventURL(), and UpnpString_get_String().

9.89.2.12 GenlibClientSubscription_get_EventURL_Length()

```
size_t GenlibClientSubscription_get_EventURL_Length (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_EventURL_Length

References GenlibClientSubscription_get_EventURL(), and UpnpString_get_Length().

9.89.2.13 GenlibClientSubscription_get_Next()

```
GenlibClientSubscription * GenlibClientSubscription_get_Next (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_Next

Referenced by freeClientSubList(), GenlibClientSubscription_assign(), GetClientSubActualSID(), GetClientSubClientSID(), and RemoveClientSubClientSID().

9.89.2.14 GenlibClientSubscription_get_RenewEventId()

```
int GenlibClientSubscription_get_RenewEventId (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_RenewEventId

Referenced by free_client_subscription(), genaRenewSubscription(), and GenlibClientSubscription_assign().

9.89.2.15 GenlibClientSubscription_get_SID()

```
const UpnpString * GenlibClientSubscription_get_SID (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_SID

Referenced by gena_process_notification_event(), genaRenewSubscription(), genaUnregisterClient(), GenlibClientSubscription_assig GenlibClientSubscription_get_SID_cstr(), GenlibClientSubscription_get_SID_Length(), and ScheduleGenaAutoRenew().

9.89.2.16 GenlibClientSubscription_get_SID_cstr()

```
const char * GenlibClientSubscription_get_SID_cstr (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_SID_cstr

References GenlibClientSubscription_get_SID(), and UpnpString_get_String().

Referenced by GetClientSubClientSID(), and RemoveClientSubClientSID().

9.89.2.17 GenlibClientSubscription_get_SID_Length()

```
size_t GenlibClientSubscription_get_SID_Length (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_SID_Length

References GenlibClientSubscription_get_SID(), and UpnpString_get_Length().

9.89.2.18 GenlibClientSubscription_new()

```
GenlibClientSubscription * GenlibClientSubscription_new ( )
```

Constructor

References UpnpString_new().

Referenced by genaRenewSubscription(), genaSubscribe(), genaUnregisterClient(), genaUnSubscribe(), and GenlibClientSubscription_dup().

9.89.2.19 GenlibClientSubscription_set_ActualSID()

```
int GenlibClientSubscription_set_ActualSID (
            GenlibClientSubscription * p,
            const UpnpString * s )
```

GenlibClientSubscription_set_ActualSID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by genaRenewSubscription(), genaSubscribe(), and GenlibClientSubscription_assign().

9.89.2.20 GenlibClientSubscription_set_EventURL()

```
int GenlibClientSubscription_set_EventURL (
            GenlibClientSubscription * p,
            const UpnpString * s )
```

GenlibClientSubscription_set_EventURL

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by genaSubscribe(), and GenlibClientSubscription_assign().

9.89.2.21 GenlibClientSubscription_set_Next()

```
int GenlibClientSubscription_set_Next (
            GenlibClientSubscription * p,
            GenlibClientSubscription * n )
```

GenlibClientSubscription_set_Next

Referenced by genaSubscribe(), GenlibClientSubscription_assign(), and RemoveClientSubClientSID().

9.89.2.22 GenlibClientSubscription_set_RenewEventId()

```
int GenlibClientSubscription_set_RenewEventId (
            GenlibClientSubscription * p,
            int n )
```

GenlibClientSubscription_set_RenewEventId

Referenced by free_client_subscription(), genaRenewSubscription(), genaSubscribe(), GenlibClientSubscription_assign(), and ScheduleGenaAutoRenew().

9.89.2.23 GenlibClientSubscription_set_SID()

```
int GenlibClientSubscription_set_SID (
            GenlibClientSubscription * p,
            const UpnpString * s )
```

GenlibClientSubscription_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by genaSubscribe(), and GenlibClientSubscription_assign().

9.89.2.24 GenlibClientSubscription_strcpy_ActualSID()

```
int GenlibClientSubscription_strcpy_ActualSID (
            GenlibClientSubscription * p,
            const char * s )
```

GenlibClientSubscription_strcpy_ActualSID

References UpnpString_set_String().

Referenced by free_client_subscription().

9.89.2.25 GenlibClientSubscription_strcpy_EventURL()

```
int GenlibClientSubscription_strcpy_EventURL (
            GenlibClientSubscription * p,
            const char * s )
```

GenlibClientSubscription_strcpy_EventURL

References UpnpString_set_String().

Referenced by free_client_subscription().

9.89.2.26 GenlibClientSubscription_strcpy_SID()

```
int GenlibClientSubscription_strcpy_SID (
            GenlibClientSubscription * p,
            const char * s )
```

GenlibClientSubscription_strcpy_SID

References UpnpString_set_String().

9.89.2.27 GenlibClientSubscription_strncpy_ActualSID()

```
int GenlibClientSubscription_strncpy_ActualSID (
            GenlibClientSubscription * p,
            const char * s,
            size_t n )
```

GenlibClientSubscription_strncpy_ActualSID

References UpnpString_set_StringN().

9.89.2.28 GenlibClientSubscription_strncpy_EventURL()

```
int GenlibClientSubscription_strncpy_EventURL (
            GenlibClientSubscription * p,
            const char * s,
            size_t n )
```

GenlibClientSubscription_strncpy_EventURL

References UpnpString_set_StringN().

9.89.2.29 GenlibClientSubscription_strncpy_SID()

```
int GenlibClientSubscription_strncpy_SID (
            GenlibClientSubscription * p,
            const char * s,
            size_t n )
```

GenlibClientSubscription_strncpy_SID

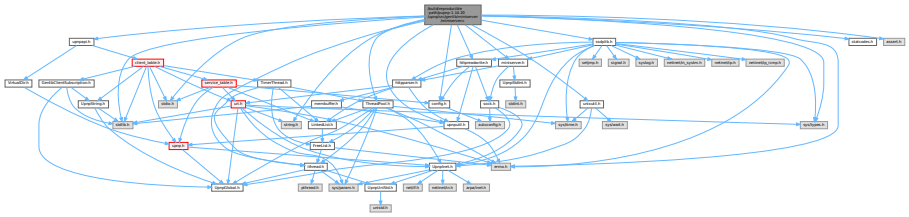
References UpnpString_set_StringN().

9.90 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/miniserver/miniserver.c File Reference

Implements the functionality and utility functions used by the Miniserver module.

```
#include "config.h"
#include "miniserver.h"
#include "ThreadPool.h"
#include "httpreadwrite.h"
#include "ithread.h"
#include "ssdplib.h"
#include "statcodes.h"
#include "unixutil.h"
#include "upnpapi.h"
#include "upnputil.h"
#include <assert.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
```

Include dependency graph for miniserver.c:



Data Structures

- struct mserv_request_t

Macros

- #define APPLICATION_LISTENING_PORT 49152

Enumerations

- enum MiniServerState { MSERV_IDLE , MSERV_RUNNING , MSERV_STOPPING }

Functions

- static UPNP_INLINE void **fdset_if_valid** (SOCKET sock, fd_set *set)
- static void **web_server_accept** (SOCKET lsock, fd_set *set)
- static void **ssdp_read** (SOCKET *rsock, fd_set *set)
- static int **receive_from_stopSock** (SOCKET ssock, fd_set *set)
- static void RunMiniServer (MiniServerSockArray *miniSock)

  Run the miniserver.
- static int get_port (SOCKET sockfd, uint16_t *port)

  Returns port to which socket, sockfd, is bound.
- static int get_miniserver_stopsock (MiniServerSockArray *out)

  Creates the miniserver STOP socket. This socket is created and listened on to know when it is time to stop the Miniserver.
- static UPNP_INLINE void **InitMiniServerSockArray** (MiniServerSockArray *miniSocket)
- int StartMiniServer (uint16_t *listen_port4, uint16_t *listen_port6, uint16_t *listen_port6UlaGua)

  Initialize the sockets functionality for the Miniserver.
- int StopMiniServer ()

  Stop and Shutdown the MiniServer and free socket resources.

Variables

- uint16_t miniStopSockPort
- static MiniServerState gMServState = MSERV_IDLE

9.90.1 Detailed Description

Implements the functionality and utility functions used by the Miniserver module.

The miniserver is a central point for processing all network requests. It is made of:

- The SSDP sockets for discovery.
- The HTTP listeners for description / control / eventing.

9.90.2 Macro Definition Documentation

9.90.2.1 APPLICATION_LISTENING_PORT

```
#define APPLICATION_LISTENING_PORT 49152
```

9.90.3 Enumeration Type Documentation

9.90.3.1 MiniServerState

```
enum MiniServerState
```

Enumerator

MSERV_IDLE	
MSERV_RUNNING	
MSERV_STOPPING	

9.90.4 Function Documentation

9.90.4.1 get_miniserver_stopsock()

```
static int get_miniserver_stopsock (
            MiniServerSockArray * out )  [static]
```

Creates the miniserver STOP socket. This socket is created and listened on to know when it is time to stop the Miniserver.

Returns

- `UPNP_E_OUTOF_SOCKET`: Failed to create a socket.
- `UPNP_E_SOCKET_BIND`: Bind() failed.
- `UPNP_E_INTERNAL_ERROR`: Port returned by the socket layer is < 0.
- `UPNP_E_SUCCESS`: Success.

Parameters

in	*out*	Miniserver Socket Array.

References get_port(), INVALID_SOCKET, MServerSockArray::miniServerStopSock, miniStopSockPort, sock_close(), SOCKET_ERROR, UPNP_E_INTERNAL_ERROR, UPNP_E_OUTOF_SOCKET, UPNP_E_SOCKET_BIND, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by StartMiniServer().

9.90.4.2 get_port()

```
static int get_port (
            SOCKET sockfd,
            uint16_t * port )  [static]
```

Returns port to which socket, sockfd, is bound.

Returns

-1 on error; check errno. 0 if successfull.

Parameters

in	*sockfd*	Socket descriptor.
out	*port*	The port value if successful, otherwise, untouched.

References UpnpPrintf().

Referenced by get_miniserver_stopsock().

9.90.4.3 RunMiniServer()

```
static void RunMiniServer (
             MiniServerSockArray * miniSock )  [static]
```

Run the miniserver.

The MiniServer accepts a new request and schedules a thread to handle the new request. Checks for socket state and invokes appropriate read and shutdown actions for the Miniserver and SSDP sockets.

Parameters

in	*miniSock*	Socket Array.

References gMServState, MServerSockArray::miniServerSock4, MServerSockArray::miniServerSock6, MServerSockArray::miniServe MServerSockArray::miniServerStopSock, MSERV_IDLE, MSERV_RUNNING, sock_close(), SOCKET_ERROR, MServerSockArray::ssdpReqSock4, MServerSockArray::ssdpReqSock6, MServerSockArray::ssdpSock4, MServerSockArray::ssdpSo MServerSockArray::ssdpSock6UlaGua, and UpnpPrintf().

Referenced by StartMiniServer().

9.90.4.4 StartMiniServer()

```
int StartMiniServer (
             uint16_t * listen_port4,
             uint16_t * listen_port6,
             uint16_t * listen_port6UlaGua )
```

Initialize the sockets functionality for the Miniserver.

Initialize a thread pool job to run the MiniServer and the job to the thread pool.

If listen port is 0, port is dynamically picked.

Use timer mechanism to start the MiniServer, failure to meet the allowed delay aborts the attempt to launch the MiniServer.

Returns

- On success: UPNP_E_SUCCESS.
- On error: UPNP_E_XXX.

Parameters

in,out	*listen_port4*	Port on which the server listens for incoming IPv4 connections.
in,out	*listen_port6*	Port on which the server listens for incoming IPv6 LLA connections.
in,out	*listen_port6UlaGua*	Port on which the server listens for incoming IPv6 ULA or GUA connections.

References get_miniserver_stopsock(), get_ssdp_sockets(), gMiniServerThreadPool, gMServState, MServerSockArray::miniServerSo
MServerSockArray::miniServerSock6, MServerSockArray::miniServerSock6UlaGua, MServerSockArray::miniServerStopSock,
MSERV_IDLE, MSERV_RUNNING, RunMiniServer(), sock_close(), MServerSockArray::ssdpReqSock4,
MServerSockArray::ssdpReqSock6, MServerSockArray::ssdpSock4, MServerSockArray::ssdpSock6, MServerSockArray::ssdpSock6
ThreadPoolAddPersistent(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_INTERNAL_ERROR,
UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by UpnpInitStartServers().

9.90.4.5 StopMiniServer()

```
int StopMiniServer ( )
```

Stop and Shutdown the MiniServer and free socket resources.

Returns

Always returns 0.

References gMServState, INVALID_SOCKET, miniStopSockPort, MSERV_IDLE, MSERV_RUNNING, MSERV_STOPPING,
sock_close(), and UpnpPrintf().

Referenced by UpnpFinish().

9.90.5 Variable Documentation

9.90.5.1 gMServState

```
MiniServerState gMServState = MSERV_IDLE  [static]
```

module vars

Referenced by RunMiniServer(), StartMiniServer(), and StopMiniServer().

9.90.5.2 miniStopSockPort

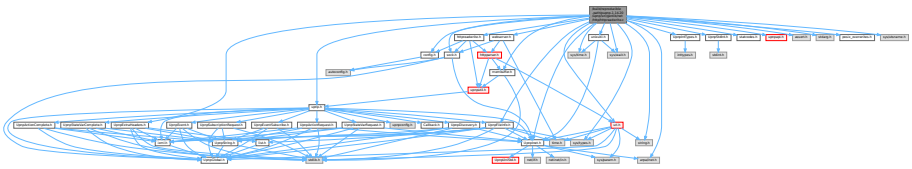
```
uint16_t miniStopSockPort
```

Referenced by get_miniserver_stopsock(), and StopMiniServer().

9.91 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/httpreadwrite.c File Reference

```
#include "config.h"
#include "httpreadwrite.h"
#include "UpnpExtraHeaders.h"
#include "UpnpFileInfo.h"
#include "UpnpInet.h"
#include "UpnpIntTypes.h"
#include "UpnpStdInt.h"
#include "membuffer.h"
#include "sock.h"
#include "statcodes.h"
#include "unixutil.h"
#include "upnp.h"
#include "upnpapi.h"
#include "uri.h"
#include "webserver.h"
#include <assert.h>
#include <stdarg.h>
#include <string.h>
#include "posix_overwrites.h"
#include <arpa/inet.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/utsname.h>
#include <sys/wait.h>
```

Include dependency graph for httpreadwrite.c:



Data Structures

- struct HTTPCONNECTIONHANDLE

Macros

- #define **CHUNK_HEADER_SIZE** (size_t)10
- #define **CHUNK_TAIL_SIZE** (size_t)10
- #define **DEFAULT_TCP_CONNECT_TIMEOUT** 5
- #define **SIZE_RANGE_BUFFER** 50

Typedefs

- typedef struct HTTPCONNECTIONHANDLE **http_connection_handle_t**

Functions

- static int Check_Connect_And_Wait_Connection (SOCKET sock, int connect_res)

  Checks socket connection and wait if it is not connected. It should be called just after connect.
- static int **private_connect** (SOCKET sockfd, const struct sockaddr *serv_addr, socklen_t addrlen)
- static int **get_hoststr** (const char *url_str, const char **hoststr, size_t *hostlen)
- static void **copy_msg_headers** (LinkedList *msgHeaders, UpnpString *headers)
- int http_FixUrl (uri_type *url, uri_type *fixed_url)

  Validates URL.
- int http_FixStrUrl (const char *urlstr, size_t urlstrlen, uri_type *fixed_url)

  Parses URL and then validates URL.
- SOCKET http_Connect (uri_type *destination_url, uri_type *url)

  Gets destination address from URL and then connects to the remote end.
- int http_RecvMessage (SOCKINFO *info, http_parser_t *parser, http_method_t request_method, int *timeout_secs, int *http_error_code)

  Get the data on the socket and take actions based on the read data to modify the parser objects buffer.
- int http_SendMessage (SOCKINFO *info, int *TimeOut, const char *fmt,...)

  Sends a message to the destination based on the format parameter.
- int **http_RequestAndResponse** (uri_type *destination, const char *request, size_t request_length, http_↩ method_t req_method, int timeout_secs, http_parser_t *response)
- int **http_Download** (const char *url_str, int timeout_secs, char **document, size_t *doc_length, char *content_type)
- int **MakeGenericMessage** (http_method_t method, const char *url_str, membuffer *request, uri_type *url, int contentLength, const char *contentType, const UpnpString *headers)
- static int ReadResponseLineAndHeaders (SOCKINFO *info, http_parser_t *parser, int *timeout_secs, int *http_error_code)

  Parses already exiting data. If not complete reads more data on the connected socket. The read data is then parsed. The same methid is carried out for headers.
- int **http_HttpGetProgress** (void *Handle, size_t *length, size_t *total)
- int **http_CancelHttpGet** (void *Handle)
- int http_OpenHttpConnection (const char *url_str, void **Handle, int timeout)

  Opens a connection to the server.
- int http_MakeHttpRequest (Upnp_HttpMethod method, const char *url_str, void *Handle, UpnpString *headers, const char *contentType, int contentLength, int timeout)

  *Makes a HTTP request using a connection previously created by **UpnpOpenHttpConnection**.*
- int http_WriteHttpRequest (void *Handle, char *buf, size_t *size, int timeout)

  *Writes the content of a HTTP request initiated by a **UpnpMakeHttpRequest** call. The end of the content should be indicated by a call to **UpnpEndHttpRequest**.*
- int http_EndHttpRequest (void *Handle, int timeout)

  *Indicates the end of a HTTP request previously made by **UpnpMakeHttpRequest**.*
- int http_GetHttpResponse (void *Handle, UpnpString *headers, char **contentType, int *contentLength, int *httpStatus, int timeout)

  *Gets the response from the server using a connection previously created by **UpnpOpenHttpConnection**.*
- int http_ReadHttpResponse (void *Handle, char *buf, size_t *size, int timeout)

  *Reads the content of a response using a connection previously created by **UpnpOpenHttpConnection**.*
- int http_CloseHttpConnection (void *Handle)

  *Closes the connection created with **UpnpOpenHttpConnection** and frees any memory associated with the connection.*
- int **http_SendStatusResponse** (SOCKINFO *info, int http_status_code, int request_major_version, int request_minor_version)
- int http_MakeMessage (membuffer *buf, int http_major_version, int http_minor_version, const char *fmt,...)

  Generate an HTTP message based on the format that is specified in the input parameters.

- void **http_CalcResponseVersion** (int request_major_vers, int request_minor_vers, int *response_major_↩ vers, int *response_minor_vers)
- int **MakeGetMessageEx** (const char *url_str, membuffer *request, uri_type *url, struct SendInstruction *p↩ RangeSpecifier)
- int **http_OpenHttpGetEx** (const char *url_str, void **Handle, char **contentType, int *contentLength, int *httpStatus, int lowRange, int highRange, int timeout)
- void **get_sdk_info** (char *info, size_t infoSize)

9.91.1 Detailed Description

Purpose: This file defines the functionality making use of the http. It defines functions to receive messages, process messages, send messages.

9.91.2 Function Documentation

9.91.2.1 Check_Connect_And_Wait_Connection()

```
static int Check_Connect_And_Wait_Connection (
            SOCKET sock,
            int connect_res )  [static]
```

Checks socket connection and wait if it is not connected. It should be called just after connect.

Returns

0 if successful, else -1.

Parameters

in	*sock*	socket.
in	*connect_res*	result of connect.

References memptr::buf, membuffer::buf, Check_Connect_And_Wait_Connection(), memptr::length, membuffer::length, http_header_t::name, sock_make_blocking(), sock_make_no_blocking(), UPNP_E_INVALID_URL, UPNP_E_SUCCESS, and http_header_t::value.

Referenced by Check_Connect_And_Wait_Connection().

9.91.2.2 http_CloseHttpConnection()

```
int http_CloseHttpConnection (
            void * handle )
```

Closes the connection created with **UpnpOpenHttpConnection** and frees any memory associated with the connection.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: **handle**, or is not a valid pointer.
- `UPNP_E_SOCKET_READ`: An error or timeout occurred reading from a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*Handle*	The handle of the connection to close, created by the call to **UpnpOpenHttpPost**.

References sock_destroy(), UPNP_E_INVALID_PARAM, and UPNP_E_SUCCESS.

Referenced by UpnpCloseHttpConnection(), and UpnpCloseHttpPost().

9.91.2.3 http_Connect()

```
SOCKET http_Connect (
            uri_type * destination_url,
            uri_type * url )
```

Gets destination address from URL and then connects to the remote end.

Returns

Socket descriptor on success, or on error:

- `UPNP_E_OUTOF_SOCKET`
- `UPNP_E_SOCKET_CONNECT`

Parameters

in	*destination_url*	URL containing destination information.
out	*url*	Fixed and corrected URL.

References http_FixUrl(), INVALID_SOCKET, UPNP_E_OUTOF_SOCKET, UPNP_E_SOCKET_CONNECT, UpnpCloseSocket, and UpnpPrintf().

Referenced by notify_send_and_recv().

9.91.2.4 http_EndHttpRequest()

```
int http_EndHttpRequest (
            void * handle,
            int timeout )
```

Indicates the end of a HTTP request previously made by **UpnpMakeHttpRequest**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: **handle** is not a valid pointer.
- `UPNP_E_OUTOF_MEMORY`: Insufficient resources exist to download this file.
- `UPNP_E_SOCKET_ERROR`: Error occured allocating a socket and resources or an error occurred binding a socket.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_SOCKET_CONNECT`: An error occurred connecting a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*Handle*	The handle to the connection.
in	*timeout*	The time out value sent with the request during which a response is expected from the receiver, failing which, an error is reported. If value is negative, timeout is infinite.

References sock_write(), UPNP_E_INVALID_PARAM, UPNP_E_SOCKET_WRITE, and UPNP_E_SUCCESS.

Referenced by UpnpCloseHttpPost(), and UpnpEndHttpRequest().

9.91.2.5 http_FixStrUrl()

```
int http_FixStrUrl (
            const char * urlstr,
            size_t urlstrlen,
            uri_type * fixed_url )
```

Parses URL and then validates URL.

Returns

- `UPNP_E_INVALID_URL`
- `UPNP_E_SUCCESS`

Parameters

in	*urlstr*	Character string as a URL.
in	*urlstrlen*	Length of the character string.
out	*fixed_url*	Fixed and corrected URL.

References http_FixUrl(), parse_uri(), and UPNP_E_INVALID_URL.

Referenced by gena_subscribe(), gena_unsubscribe(), and http_OpenHttpConnection().

9.91.2.6 http_FixUrl()

```
int http_FixUrl (
```

```
            uri_type * url,
            uri_type * fixed_url )
```

Validates URL.

Returns

- UPNP_E_INVALID_URL
- UPNP_E_SUCCESS

Parameters

in	*url*	URL to be validated and fixed.
out	*fixed_url*	URL after being fixed.

References HOSTPORT::text, token_string_casecmp(), UPNP_E_INVALID_URL, and UPNP_E_SUCCESS.

Referenced by http_Connect(), http_FixStrUrl(), and http_MakeMessage().

9.91.2.7 http_GetHttpResponse()

```
int http_GetHttpResponse (
            void * handle,
            UpnpString * headers,
            char ** contentType,
            int * contentLength,
            int * httpStatus,
            int timeout )
```

Gets the response from the server using a connection previously created by **UpnpOpenHttpConnection**.

Note

Memory for **contentType** is only valid until the next call to the HTTP API for the same connection.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: Either **handle**, **contentType**, **contentLength** or **httpStatus** is not a valid pointer.
- UPNP_E_INVALID_URL: The **url** is not a valid URL.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to download this file.
- UPNP_E_NETWORK_ERROR: A network error occurred.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_READ: An error or timeout occurred reading from a socket.
- UPNP_E_SOCKET_BIND: An error occurred binding a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting a socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.
- UPNP_E_BAD_RESPONSE: A bad response was received from the remote server.

Parameters

in	*Handle*	The handle of the connection created by the call to **UpnpOpenHttpConnection**.
in	*headers*	Headers sent by the server for the response. If NULL then the headers are not copied.
out	*contentType*	A buffer to store the media type of the item.
out	*contentLength*	A pointer to store the length of the item.
out	*httpStatus*	The status returned on receiving a response message.
in	*timeout*	The time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is negative, timeout is infinite.

References memptr::buf, http_message_t::headers, PARSE_CONTINUE_1, PARSE_OK, PARSE_SUCCESS, ReadResponseLineAndHeaders(), http_message_t::status_code, UPNP_E_BAD_RESPONSE, and UPNP_E_SUCCESS.

Referenced by UpnpCloseHttpPost(), and UpnpGetHttpResponse().

9.91.2.8 http_MakeHttpRequest()

```
int http_MakeHttpRequest (
            Upnp_HttpMethod method,
            const char * url,
            void * handle,
            UpnpString * headers,
            const char * contentType,
            int contentLength,
            int timeout )
```

Makes a HTTP request using a connection previously created by **UpnpOpenHttpConnection**.

Note

Trying to make another request while a request is already being processed results in undefined behavior. It's up to the user to end a previous request by calling **UpnpEndHttpRequest**.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: Either **url**, **handle** or **contentType** is not a valid pointer.
- UPNP_E_INVALID_URL: The **url** is not a valid URL.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to download this file.
- UPNP_E_SOCKET_ERROR: Error occured allocating a socket and resources or an error occurred binding a socket.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting a socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.

Parameters

in	*url_str*	The URL to use to make the request. The URL should use the same host and scheme used to create the connection.
in	*Handle*	The handle to the connection.
in	*headers*	Headers to be used for the request. Each header should be terminated by a CRLF as specified in the HTTP specification. If NULL then the default headers will be used.
in	*contentType*	The media type of content being sent. Can be NULL.
in	*contentLength*	The length of the content being sent, in bytes. Set to **UPNP_USING_CHUNKED** to use chunked encoding, or **UPNP_UNTIL_CLOSE** to avoid specifying the content length to the server. In this case the request is considered unfinished until the connection is closed.
in	*timeout*	The time out value sent with the request during which a response is expected from the receiver, failing which, an error is reported. If value is negative, timeout is infinite.

References membuffer::buf, http_SendMessage(), membuffer::length, UPNP_E_INVALID_PARAM, and UPNP_E_SUCCESS.

Referenced by UpnpMakeHttpRequest(), and UpnpOpenHttpPost().

9.91.2.9 http_MakeMessage()

```
int http_MakeMessage (
            membuffer * buf,
            int http_major_version,
            int http_minor_version,
            const char * fmt,
             ... )
```

Generate an HTTP message based on the format that is specified in the input parameters.

```
Format types:
        'B':    arg = int status_code           -- appends content-length,
content-type and HTML body for given code. 'b': arg1 = const char *buf; arg2 =
size_t buf_length memory ptr
        'C':    (no args)                       -- appends a HTTP CONNECTION:
close header depending on major, minor version. 'c':    (no args)
-- appends CRLF "\r\n"
        'D':    (no args)                       -- appends HTTP DATE: header
        'd':    arg = int number                -- appends decimal number
        'G':    arg = range information         -- add range header
        'h':    arg = off_t number              -- appends off_t number
        'K':    (no args)                       -- add chunky header
        'L':    arg = language information      -- add Content-Language header
if Accept-Language header is not empty and if WEB_SERVER_CONTENT_LANGUAGE is not
empty 'N':      arg1 = off_t content_length     -- content-length header 'q':
arg1 = http_method_t            -- request start line and HOST header arg2 =
(uri_type *) 'Q':       arg1 = http_method_t;           -- start line of request
                arg2 = char* url;
                arg3 = size_t url_length
        'R':    arg = int status_code           -- adds a response start line
        'S':    (no args)                       -- appends HTTP SERVER: header
        's':    arg = const char *              -- C_string
        'T':    arg = char * content_type;      -- format e.g: "text/html";
content-type header
        't':    arg = time_t * gmt_time         -- appends time in RFC 1123 fmt
        'U':    (no args)                       -- appends HTTP USER-AGENT:
header
        'X':    arg = const char *              -- useragent; "redsonic" HTTP
X-User-Agent: useragent
```

Returns

- 0 - On Success
- UPNP_E_OUTOF_MEMORY
- UPNP_E_INVALID_URL

References SendInstruction::CorsHeader, DOMString, http_FixUrl(), http_MakeMessage(), HOSTPORT::text, UPNP_E_INVALID_URL, UPNP_E_OUTOF_MEMORY, UpnpExtraHeaders_get_resp(), and UpnpPrintf().

Referenced by CreateServicePacket(), gena_subscribe(), gena_unsubscribe(), genaNotify(), http_MakeMessage(), notify_send_and_recv(), process_request(), respond_ok(), send_action_response(), send_error_response(), send_var_query_response(), and web_server_callback().

9.91.2.10 http_OpenHttpConnection()

```
int http_OpenHttpConnection (
            const char * url,
            void ** handle,
            int timeout )
```

Opens a connection to the server.

The SDK allocates the memory for **handle**, the application is responsible for freeing this memory.

Returns

An integer representing one of the following:

- UPNP_E_SUCCESS: The operation completed successfully.
- UPNP_E_INVALID_PARAM: Either **url**, or **handle** is not a valid pointer.
- UPNP_E_INVALID_URL: The **url** is not a valid URL.
- UPNP_E_OUTOF_MEMORY: Insufficient resources exist to download this file.
- UPNP_E_SOCKET_ERROR: Error occured allocating a socket and resources or an error occurred binding a socket.
- UPNP_E_SOCKET_WRITE: An error or timeout occurred writing to a socket.
- UPNP_E_SOCKET_CONNECT: An error occurred connecting a socket.
- UPNP_E_OUTOF_SOCKET: Too many sockets are currently allocated.

Parameters

in	*url_str*	The URL which contains the host, and the scheme to make the connection.
in,out	*Handle*	A pointer in which to store the handle for this connection. This handle is required for futher operations over this connection.
in	*timeout*	The time out value sent with the request during which a response is expected from the receiver, failing which, an error is reported. If value is negative, timeout is infinite.

References http_FixStrUrl(), INVALID_SOCKET, sock_destroy(), sock_init(), token_string_casecmp(), UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, UPNP_E_SOCKET_CONNECT, UPNP_E_SOCKET_ERROR, and UPNP_E_SUCCESS.

Referenced by UpnpOpenHttpConnection(), and UpnpOpenHttpPost().

9.91.2.11 http_ReadHttpResponse()

```
int http_ReadHttpResponse (
            void * handle,
            char * buf,
            size_t * size,
            int timeout )
```

Reads the content of a response using a connection previously created by **UpnpOpenHttpConnection**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, **buf** or **size** is not a valid pointer.
- `UPNP_E_BAD_RESPONSE`: A bad response was received from the remote server.
- `UPNP_E_BAD_HTTPMSG`: Either the request or response was in the incorrect format.
- `UPNP_E_CANCELED`: another thread called UpnpCancelHttpGet.

Note: In case of return values, the status code parameter of the passed in handle value may provide additional information on the return value.

Parameters

`in`	*Handle*	The handle of the connection created by the call to **UpnpOpenHttpConnection**.
`in,out`	*buf*	The buffer to store the read item.
`in,out`	*size*	The size of the buffer to be read.
`in`	*timeout*	The time out value sent with the request during which a response is expected from the server, failing which, an error is reported back to the user. If value is negative, timeout is infinite.

References http_message_t::amount_discarded, membuffer::buf, scanner_t::cursor, http_message_t::entity, http_parser_t::entity_start_position, http_parser_t::http_error_code, memptr::length, http_message_t::msg, PARSE_CONTINUE_1, PARSE_FAILURE, PARSE_INCOMPLETE, PARSE_INCOMPLETE_ENTITY, PARSE_SUCCESS, sock_read(), UPNP_E_BAD_HTTPMSG, UPNP_E_BAD_RESPONSE, UPNP_E_CANCELED, UPNP_E_INVALID_PARAM, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by UpnpReadHttpGet(), and UpnpReadHttpResponse().

9.91.2.12 http_RecvMessage()

```
int http_RecvMessage (
            SOCKINFO * info,
            http_parser_t * parser,
            http_method_t request_method,
            int * timeout_secs,
            int * http_error_code )
```

Get the data on the socket and take actions based on the read data to modify the parser objects buffer.

If an error is reported while parsing the data, the error code is passed in the http_errr_code parameter.

Parameters: IN SOCKINFO *info; Socket information object OUT http_parser_t* parser; HTTP parser object IN http↩ _method_t request_method; HTTP request method IN OUT int* timeout_secs; time out OUT int* http_error_code; HTTP error code returned

Returns

UPNP_E_SUCCESS UPNP_E_BAD_HTTPMSG

References membuffer::buf, g_maxContentLength, http_parser_t::http_error_code, http_message_t::msg, PARSE_CONTINUE_1, PARSE_FAILURE, PARSE_INCOMPLETE_ENTITY, PARSE_NO_MATCH, PARSE_SUCCESS, sock_read(), UPNP_E_BAD_HTTPMSG, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by notify_send_and_recv().

9.91.2.13 http_SendMessage()

```
int http_SendMessage (
             SOCKINFO * info,
             int * timeout_secs,
             const char * fmt,
              ...  )
```

Sends a message to the destination based on the format parameter.

fmt types:

- `'f'`: arg = "const char *" file name
- `'b'`: arg1 = "const char *" mem_buffer; arg2 = "size_t" buffer length.
- `'I'`: arg = "struct SendInstruction *"

E.g.:

```
char *buf = "POST /xyz.cgi http/1.1\r\n\r\n";
char *filename = "foo.dat";
int status = http_SendMessage(tcpsock, "bf",
 buf, strlen(buf),       // args for memory buffer
 filename);              // arg for file
```

Returns

- UPNP_E_OUTOF_MEMORY
- UPNP_E_FILE_READ_ERROR
- UPNP_E_SUCCESS

References VirtualDirCallbacks::close, SendInstruction::Cookie, VirtualDirCallbacks::open, PRIzd, VirtualDirCallbacks::read, SendInstruction::ReadSendSize, SendInstruction::RequestCookie, VirtualDirCallbacks::seek, sock_write(), UPNP_E_FILE_READ_ERROR, UPNP_E_INTERNAL_ERROR, UPNP_E_OUTOF_MEMORY, UpnpPrintf(), and virtualDirCallback.

Referenced by http_MakeHttpRequest(), notify_send_and_recv(), respond_ok(), send_action_response(), send_error_response(), send_var_query_response(), and web_server_callback().

9.91.2.14 http_WriteHttpRequest()

```
int http_WriteHttpRequest (
            void * handle,
            char * buf,
            size_t * size,
            int timeout )
```

Writes the content of a HTTP request initiated by a **UpnpMakeHttpRequest** call. The end of the content should be indicated by a call to **UpnpEndHttpRequest**.

Returns

An integer representing one of the following:

- `UPNP_E_SUCCESS`: The operation completed successfully.
- `UPNP_E_INVALID_PARAM`: Either **handle**, **buf** or **size** is not a valid pointer.
- `UPNP_E_SOCKET_WRITE`: An error or timeout occurred writing to a socket.
- `UPNP_E_OUTOF_SOCKET`: Too many sockets are currently allocated.

Parameters

in	*Handle*	The handle of the connection created by the call to **UpnpOpenHttpConnection**.
in	*buf*	The buffer containing date to be written.
in	*size*	The size, in bytes of **buf**.
in	*timeout*	A timeout value sent with the request during which a response is expected from the server, failing which, an error is reported. If value is negative, timeout is infinite.

References sock_write(), UPNP_E_INVALID_PARAM, UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by UpnpWriteHttpPost(), and UpnpWriteHttpRequest().

9.91.2.15 ReadResponseLineAndHeaders()

```
static int ReadResponseLineAndHeaders (
            SOCKINFO * info,
            http_parser_t * parser,
            int * timeout_secs,
            int * http_error_code )  [static]
```

Parses already exiting data. If not complete reads more data on the connected socket. The read data is then parsed. The same methid is carried out for headers.

Returns

integer:

- `PARSE_OK` - On Success
- `PARSE_FAILURE` - Failure to parse data correctly
- `UPNP_E_BAD_HTTPMSG` - Socker read() returns an error

Parameters

info	Socket information object.
parser	HTTP Parser object.
timeout_secs	Time out value.
http_error_code	HTTP errror code returned.

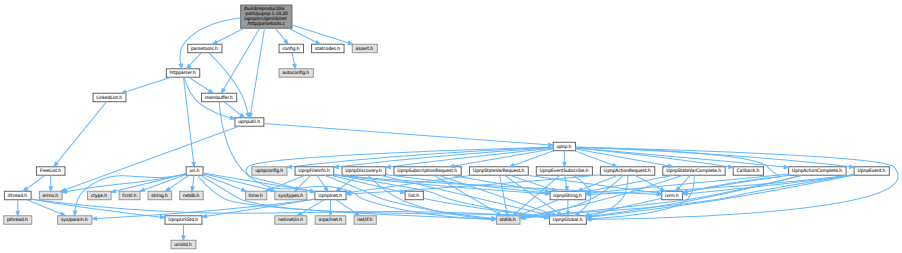
References http_parser_t::http_error_code, http_message_t::msg, PARSE_FAILURE, PARSE_INCOMPLETE, PARSE_OK, sock_read(), and UPNP_E_BAD_HTTPMSG.

Referenced by http_GetHttpResponse().

9.92 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/parsetools.c File Reference

```
#include "parsetools.h"
#include "config.h"
#include "httpparser.h"
#include "membuffer.h"
#include "statcodes.h"
#include "upnputil.h"
#include <assert.h>
```

Include dependency graph for parsetools.c:



Functions

- int has_xml_content_type (http_message_t *hmsg)

  Find the header from the HTTP message and match the header for xml data.

9.92.1 Detailed Description

Purpose: This file a function to extract the header information from an http message and then matches the data with XML data.

9.92.2 Function Documentation

9.92.2.1 has_xml_content_type()

```
int has_xml_content_type (
            http_message_t * hmsg )
```

Find the header from the HTTP message and match the header for xml data.

Returns

boolean.

Parameters

in	*hmsg*	HTTP Message object.

References memptr::buf, memptr::length, and PARSE_OK.

Referenced by gena_process_notification_event(), and soap_device_callback().

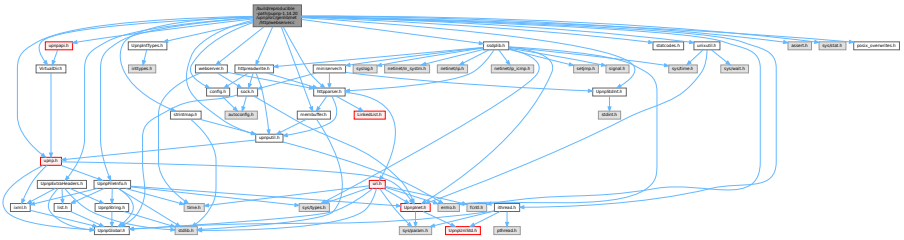
9.93 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/webserver.c File Reference

Defines the Web Server and has functions to carry out operations of the Web Server.

```
#include "config.h"
#include "webserver.h"
#include "UpnpExtraHeaders.h"
#include "UpnpFileInfo.h"
#include "UpnpIntTypes.h"
#include "UpnpStdInt.h"
#include "VirtualDir.h"
#include "httpparser.h"
#include "httpreadwrite.h"
#include "ithread.h"
#include "membuffer.h"
#include "ssdplib.h"
#include "statcodes.h"
#include "strintmap.h"
#include "unixutil.h"
#include "upnp.h"
#include "upnpapi.h"
#include "upnputil.h"
#include <assert.h>
#include <fcntl.h>
#include <sys/stat.h>
```

```
#include "posix_overwrites.h"
```

Include dependency graph for webserver.c:



Data Structures

- struct document_type_t
- struct xml_alias_t

Macros

- #define **AUDIO_STR** "\1"
- #define **VIDEO_STR** "\2"
- #define **IMAGE_STR** "\3"
- #define **APPLICATION_STR** "\4"
- #define **TEXT_STR** "\5"
- #define **APPLICATION_INDEX** 4
- #define **TEXT_INDEX** 5
- #define **NUM_MEDIA_TYPES** 70
- #define **NUM_HTTP_HEADER_NAMES** 33
- #define **ASCTIME_R_BUFFER_SIZE** 26
- #define **web_server_asctime_r** asctime_r

Enumerations

- enum resp_type {
  RESP_FILEDOC , **RESP_XMLDOC** , **RESP_HEADERS** , **RESP_WEBDOC** ,
  RESP_POST }

Functions

- static UPNP_INLINE void media_list_init (void)

  Decodes list and stores it in gMediaTypeList.
- static UPNP_INLINE int search_extension (const char *extension, const char **con_type, const char **con_subtype)

  Based on the extension, returns the content type and content subtype.
- static UPNP_INLINE int get_content_type (const char *filename, UpnpFileInfo *fileInfo)

  Based on the extension, clones an XML string based on type and content subtype. If content type and sub type are not found, unknown types are used.
- static UPNP_INLINE void glob_alias_init (void)

  Initialize the global XML document. Allocate buffers for the XML document.
- static UPNP_INLINE int is_valid_alias (const struct xml_alias_t *alias)

Check for the validity of the XML object buffer.

- static void alias_grab (struct xml_alias_t *alias)

  Copy the contents of the global XML document into the local output parameter.
- static void alias_release (struct xml_alias_t *alias)

  Release the XML document referred to by the input parameter. Free the allocated buffers associated with this object.
- int web_server_set_alias (const char *alias_name, const char *alias_content, size_t alias_content_length, time_t last_modified)

  Replaces current alias with the given alias. To remove the current alias, set alias_name to NULL.
- int web_server_init ()

  Initilialize the different documents. Initialize the memory for root directory for web server. Call to initialize global XML document. Sets bWebServerState to WEB_SERVER_ENABLED.
- void web_server_destroy (void)

  Release memory allocated for the global web server root directory and the global XML document. Resets the flag bWebServerState to WEB_SERVER_DISABLED.
- static int get_file_info (const char *filename, UpnpFileInfo *info)

  Release memory allocated for the global web server root directory and the global XML document. Resets the flag bWebServerState to WEB_SERVER_DISABLED.
- int web_server_set_root_dir (const char *root_dir)

  Assign the path specfied by the input const char root_dir parameter to the global Document root directory. Also check for path names ending in '/'.*
- int web_server_set_cors (const char *cors_string)

  Assign the Access-Control-Allow-Origin specfied by the input const char cors_string parameterto the global CORS string.*
- static UPNP_INLINE int get_alias (const char *request_file, struct xml_alias_t *alias, UpnpFileInfo *info)

  Compare the files names between the one on the XML alias the one passed in as the input parameter. If equal extract file information.
- static int isFileInVirtualDir (char *filePath, const void **cookie)

  Compares filePath with paths from the list of virtual directory lists.
- static void ToUpperCase (char *s)

  Converts input string to upper case.
- static char * StrStr (char *s1, const char *s2)

  Finds a substring from a string in a case insensitive way.
- static char * StrTok (char **Src, const char *Del)

  Finds next token in a string.
- static int GetNextRange (char **SrcRangeStr, off_t *FirstByte, off_t *LastByte)

  Returns a range of integers from a string.
- static int CreateHTTPRangeResponseHeader (char *ByteRangeSpecifier, off_t FileLength, struct SendInstruction *Instr)

  Fills in the Offset, read size and contents to send out as an HTTP Range Response.
- static int CheckOtherHTTPHeaders (http_message_t *Req, struct SendInstruction *RespInstr, off_t FileSize)

  Get header id from the request parameter and take appropriate action based on the ids as an HTTP Range Response.
- static void FreeExtraHTTPHeaders (UpnpListHead *extraHeadersList)
- static int ExtraHTTPHeaders (http_message_t *Req, UpnpListHead *extraHeadersList)

  Build an array of unrecognized headers.
- static int process_request (SOCKINFO *info, http_message_t *req, enum resp_type *rtype, membuffer *headers, membuffer *filename, struct xml_alias_t *alias, struct SendInstruction *RespInstr)

  Processes the request and returns the result in the output parameters.
- static int http_RecvPostMessage (http_parser_t *parser, SOCKINFO *info, char *filename, struct SendInstruction *Instr)

  Receives the HTTP post message.
- void web_server_callback (http_parser_t *parser, http_message_t *req, SOCKINFO *info)

  Main entry point into web server; Handles HTTP GET and HEAD requests.

Variables

- static const char * gMediaTypes [ ]
- static const char * **gEncodedMediaTypes**
- static struct document_type_t gMediaTypeList [70]
- membuffer gDocumentRootDir
- membuffer gWebserverCorsString
- static struct xml_alias_t gAliasDoc
- static ithread_mutex_t **gWebMutex**
- str_int_entry **Http_Header_Names** [33]

9.93.1 Detailed Description

Defines the Web Server and has functions to carry out operations of the Web Server.

9.93.2 Enumeration Type Documentation

9.93.2.1 resp_type

```
enum resp_type
```

Response Types.

9.93.3 Function Documentation

9.93.3.1 alias_grab()

```
static void alias_grab (
            struct xml_alias_t * alias )  [static]
```

Copy the contents of the global XML document into the local output parameter.

Parameters

out	*alias*	XML alias object.

References xml_alias_t::ct, gAliasDoc, and is_valid_alias().

Referenced by process_request().

9.93.3.2 alias_release()

```
static void alias_release (
            struct xml_alias_t * alias )  [static]
```

Release the XML document referred to by the input parameter. Free the allocated buffers associated with this object.

Parameters

in	*alias*	XML alias object.

References xml_alias_t::ct, xml_alias_t::doc, is_valid_alias(), and xml_alias_t::name.

Referenced by process_request(), web_server_callback(), web_server_destroy(), and web_server_set_alias().

9.93.3.3 CheckOtherHTTPHeaders()

```
static int CheckOtherHTTPHeaders (
            http_message_t * Req,
            struct SendInstruction * RespInstr,
            off_t FileSize )  [static]
```

Get header id from the request parameter and take appropriate action based on the ids as an HTTP Range Response.

Returns

- HTTP_BAD_REQUEST
- HTTP_INTERNAL_SERVER_ERROR
- HTTP_REQUEST_RANGE_NOT_SATISFIABLE
- HTTP_OK

Parameters

in	*Req*	HTTP Request message.
out	*RespInstr*	Send Instruction object to data for the response.
	FileSize	Size of the file containing the request document.

References memptr::buf, membuffer::buf, CreateHTTPRangeResponseHeader(), http_message_t::headers, memptr::length, membuffer::length, http_header_t::name, SendInstruction::RecvWriteSize, StrStr(), and http_header_t::value.

Referenced by process_request().

9.93.3.4 CreateHTTPRangeResponseHeader()

```
static int CreateHTTPRangeResponseHeader (
            char * ByteRangeSpecifier,
            off_t FileLength,
            struct SendInstruction * Instr )  [static]
```

Fills in the Offset, read size and contents to send out as an HTTP Range Response.

Returns

- HTTP_BAD_REQUEST
- HTTP_INTERNAL_SERVER_ERROR
- HTTP_REQUEST_RANGE_NOT_SATISFIABLE
- HTTP_OK

Parameters

	ByteRangeSpecifier	String containing the range.
	FileLength	Length of the file.
out	*Instr*	SendInstruction object where the range operations will be stored.

References GetNextRange(), SendInstruction::ReadSendSize, and StrStr().

Referenced by CheckOtherHTTPHeaders().

9.93.3.5 ExtraHTTPHeaders()

```
static int ExtraHTTPHeaders (
            http_message_t * Req,
            UpnpListHead * extraHeadersList )  [static]
```

Build an array of unrecognized headers.

Returns

nothing

Parameters

in	*Req*	HTTP Request message.

References memptr::buf, membuffer::buf, FreeExtraHTTPHeaders(), http_message_t::headers, memptr::length, membuffer::length, http_header_t::name, UpnpExtraHeaders_get_node(), UpnpExtraHeaders_new(), UpnpExtraHeaders_strncpy_na UpnpExtraHeaders_strncpy_value(), and http_header_t::value.

Referenced by process_request().

9.93.3.6 FreeExtraHTTPHeaders()

```
static void FreeExtraHTTPHeaders (
            UpnpListHead * extraHeadersList )  [static]
```

Parameters

in	*extraHeadersList*	extra HTTP headers to free.

References UpnpExtraHeaders_delete().

Referenced by ExtraHTTPHeaders(), and process_request().

9.93.3.7 get_alias()

```
static UPNP_INLINE int get_alias (
            const char * request_file,
```

```
            struct xml_alias_t * alias,
            UpnpFileInfo * info )  [static]
```

Compare the files names between the one on the XML alias the one passed in as the input parameter. If equal extract file information.

Returns

- 1 - On Success
- 0 if request is not an alias

Parameters

in	*request_file*	request file passed in to be compared with.
out	*alias*	xml alias object which has a file name stored.
out	*info*	File information object which will be filled up if the file comparison succeeds.

References membuffer::buf, xml_alias_t::doc, xml_alias_t::last_modified, membuffer::length, xml_alias_t::name, UpnpFileInfo_set_FileLength(), UpnpFileInfo_set_IsDirectory(), UpnpFileInfo_set_IsReadable(), and UpnpFileInfo_set_LastModified().

Referenced by process_request().

9.93.3.8 get_content_type()

```
static UPNP_INLINE int get_content_type (
            const char * filename,
            UpnpFileInfo * fileInfo )  [static]
```

Based on the extension, clones an XML string based on type and content subtype. If content type and sub type are not found, unknown types are used.

Returns

- 0 on success.
- UPNP_E_OUTOF_MEMORY - on memory allocation failures.

Parameters

in	*filename*	.
out	*fileInfo*	.

References search_extension(), UPNP_E_OUTOF_MEMORY, UpnpFileInfo_get_ContentType(), and UpnpFileInfo_set_ContentType().

Referenced by get_file_info().

9.93.3.9 get_file_info()

```
static int get_file_info (
            const char * filename,
            UpnpFileInfo * info )  [static]
```

Release memory allocated for the global web server root directory and the global XML document. Resets the flag bWebServerState to WEB_SERVER_DISABLED.

Returns

Integer.

Parameters

in	*filename*	Filename having the description document.
out	*info*	File information object having file attributes such as filelength, when was the file last modified, whether a file or a directory and whether the file or directory is readable.

References get_content_type(), UpnpFileInfo_get_FileLength(), UpnpFileInfo_get_IsReadable(), UpnpFileInfo_get_LastModified(), UpnpFileInfo_set_ContentType(), UpnpFileInfo_set_FileLength(), UpnpFileInfo_set_IsDirectory(), UpnpFileInfo_set_IsReadable(), UpnpFileInfo_set_LastModified(), and UpnpPrintf().

Referenced by process_request().

9.93.3.10 GetNextRange()

```
static int GetNextRange (
            char ** SrcRangeStr,
            off_t * FirstByte,
            off_t * LastByte )  [static]
```

Returns a range of integers from a string.

Returns

Always returns 1.

Parameters

SrcRangeStr	string containing the token / range.
FirstByte	gets the first byte of the token.
LastByte	gets the last byte of the token.

References StrTok().

Referenced by CreateHTTPRangeResponseHeader().

9.93.3.11 glob_alias_init()

```
static UPNP_INLINE void glob_alias_init (
            void  )  [static]
```

Initialize the global XML document. Allocate buffers for the XML document.

References xml_alias_t::ct, xml_alias_t::doc, gAliasDoc, xml_alias_t::last_modified, and xml_alias_t::name.

Referenced by web_server_init().

9.93.3.12 http_RecvPostMessage()

```
static int http_RecvPostMessage (
            http_parser_t * parser,
            SOCKINFO * info,
            char * filename,
            struct SendInstruction * Instr )  [static]
```

Receives the HTTP post message.

Returns

- HTTP_INTERNAL_SERVER_ERROR
- HTTP_UNAUTHORIZED
- HTTP_BAD_REQUEST
- HTTP_SERVICE_UNAVAILABLE
- HTTP_OK

Parameters

	parser	HTTP Parser object.
in	*info*	Socket Information object.
	filename	File where received data is copied to.
	Instr	Send Instruction object which gives information whether the file is a virtual file or not.

References membuffer::buf, VirtualDirCallbacks::close, SendInstruction::Cookie, http_message_t::entity, http_parser_t::entity_start_position, http_parser_t::http_error_code, memptr::length, http_message_t::msg, VirtualDirCallbacks::open, PARSE_CONTINUE_1, PARSE_INCOMPLETE, PARSE_INCOMPLETE_ENTITY, PARSE_OK, PARSE_SUCCESS, SendInstruction::RequestCookie, sock_read(), UpnpPrintf(), virtualDirCallback, and VirtualDirCallbacks::write.

Referenced by web_server_callback().

9.93.3.13 is_valid_alias()

```
static UPNP_INLINE int is_valid_alias (
            const struct xml_alias_t * alias )  [static]
```

Check for the validity of the XML object buffer.

Returns

int.

Parameters

in	*alias*	XML alias object.

References membuffer::buf, and xml_alias_t::doc.

Referenced by alias_grab(), alias_release(), and process_request().

9.93.3.14 isFileInVirtualDir()

```
static int isFileInVirtualDir (
            char * filePath,
            const void ** cookie )  [static]
```

Compares filePath with paths from the list of virtual directory lists.

Returns

int.

Parameters

in	*filePath*	Directory path to be tested for virtual directory.
out	*cookie*	The cookie registered with this virtual directory, if matched.

References pVirtualDirList.

Referenced by process_request().

9.93.3.15 media_list_init()

```
static UPNP_INLINE void media_list_init (
            void  )  [static]
```

Decodes list and stores it in gMediaTypeList.

References document_type_t::content_subtype, document_type_t::content_type, document_type_t::file_ext, and gMediaTypeList.

Referenced by web_server_init().

9.93.3.16 process_request()

```
static int process_request (
            SOCKINFO * info,
            http_message_t * req,
            enum resp_type * rtype,
            membuffer * headers,
            membuffer * filename,
            struct xml_alias_t * alias,
            struct SendInstruction * RespInstr )  [static]
```

Processes the request and returns the result in the output parameters.

Returns

- HTTP_BAD_REQUEST
- HTTP_INTERNAL_SERVER_ERROR
- HTTP_REQUEST_RANGE_NOT_SATISFIABLE
- HTTP_FORBIDDEN
- HTTP_NOT_FOUND
- HTTP_NOT_ACCEPTABLE
- HTTP_OK

Parameters

in	*info*	Socket info.
in	*req*	HTTP Request message.
out	*rtype*	Tpye of response.
out	*headers*	Headers.
out	*filename*	Get filename from request document.
out	*alias*	Xml alias document from the request document.
out	*RespInstr*	Send Instruction object where the response is set up.

References alias_grab(), alias_release(), memptr::buf, membuffer::buf, CheckOtherHTTPHeaders(), SendInstruction::Cookie, SendInstruction::CorsHeader, ExtraHTTPHeaders(), SOCKINFO::foreign_sockaddr, FreeExtraHTTPHeaders(), gAliasDoc, gDocumentRootDir, get_alias(), get_file_info(), VirtualDirCallbacks::get_info, gWebserverCorsString, http_MakeMessage(), is_valid_alias(), isFileInVirtualDir(), memptr::length, membuffer::length, http_message_t::method, SendInstruction::ReadSendSize, remove_dots(), remove_escaped_chars(), SendInstruction::RequestCookie, UPNP_E_SUCCESS, UpnpFileInfo_delete(), UpnpFileInfo_get_ContentType(), UpnpFileInfo_get_ExtraHeadersList(), UpnpFileInfo_get_FileLength(), UpnpFileInfo_get_IsDirectory(), UpnpFileInfo_get_IsReadable(), UpnpFileInfo_get_LastModified(), UpnpFileInfo_new(), UpnpFileInfo_set_ContentType(), UpnpFileInfo_set_CtrlPtIPAddr(), UpnpFileInfo_strncpy_Os(), http_message_t::uri, and virtualDirCallback.

Referenced by web_server_callback().

9.93.3.17 search_extension()

```
static UPNP_INLINE int search_extension (
            const char * extension,
            const char ** con_type,
            const char ** con_subtype )  [static]
```

Based on the extension, returns the content type and content subtype.

Returns

- 0 on success
- -1 on error

Parameters

in	*extension*	.
out	*con_type*	.
out	*con_subtype*	.

References document_type_t::content_subtype, document_type_t::content_type, document_type_t::file_ext, and gMediaTypeList.

Referenced by get_content_type().

9.93.3.18 StrStr()

```
static char * StrStr (
            char * s1,
            const char * s2 )  [static]
```

Finds a substring from a string in a case insensitive way.

Returns

A pointer to the first occurence of s2 in s1.

Parameters

s1	Input string.
s2	Input sub-string.

References ToUpperCase().

Referenced by CheckOtherHTTPHeaders(), and CreateHTTPRangeResponseHeader().

9.93.3.19 StrTok()

```
static char * StrTok (
            char ** Src,
            const char * Del )  [static]
```

Finds next token in a string.

Returns

Pointer to the next token.

Parameters

Src	String containing the token.
Del	Set of delimiter characters.

Referenced by GetNextRange().

9.93.3.20 ToUpperCase()

```
static void ToUpperCase (
            char * s )  [static]
```

Converts input string to upper case.

Parameters

s	Input string to be converted.

Referenced by StrStr().

9.93.3.21 web_server_callback()

```
void web_server_callback (
            http_parser_t * parser,
            http_message_t * req,
            SOCKINFO * info )
```

Main entry point into web server; Handles HTTP GET and HEAD requests.

Parameters

in	*parser*	.
in	*req*	.
in,out	*info*	.

References alias_release(), membuffer::buf, xml_alias_t::doc, http_MakeMessage(), http_RecvPostMessage(), http_SendMessage(), membuffer::length, process_request(), and UpnpPrintf().

Referenced by UpnpEnableWebserver().

9.93.3.22 web_server_destroy()

```
void web_server_destroy (
            void  )
```

Release memory allocated for the global web server root directory and the global XML document. Resets the flag bWebServerState to WEB_SERVER_DISABLED.

References alias_release(), bWebServerState, gAliasDoc, gDocumentRootDir, and gWebserverCorsString.

Referenced by UpnpEnableWebserver(), and UpnpFinish().

9.93.3.23 web_server_init()

```
int web_server_init (
            void  )
```

Initilialize the different documents. Initialize the memory for root directory for web server. Call to initialize global XML document. Sets bWebServerState to WEB_SERVER_ENABLED.

Note

alias_content is not freed here

Returns

- 0 - OK
- UPNP_E_OUTOF_MEMORY

References bWebServerState, VirtualDirCallbacks::close, gDocumentRootDir, VirtualDirCallbacks::get_info, glob_alias_init(), gWebserverCorsString, media_list_init(), VirtualDirCallbacks::open, pVirtualDirList, VirtualDirCallbacks::read, VirtualDirCallbacks::seek, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, virtualDirCallback, and VirtualDirCallbacks::write.

Referenced by UpnpEnableWebserver().

9.93.3.24 web_server_set_alias()

```
int web_server_set_alias (
            const char * alias_name,
            const char * alias_content,
            size_t alias_content_length,
            time_t last_modified )
```

Replaces current alias with the given alias. To remove the current alias, set alias_name to NULL.

Note

alias_content is not freed here

Returns

- 0 - OK
- UPNP_E_OUTOF_MEMORY

Parameters

in	*alias_name*	Webserver name of alias; created by caller and freed by caller (doesn't even have to be malloc()d.
in	*alias_content*	The xml doc; this is allocated by the caller; and freed by the web server.
in	*alias_content_length*	Length of alias body in bytes.
in	*last_modified*	Time when the contents of alias were last changed (local time).

References alias_release(), xml_alias_t::ct, xml_alias_t::doc, gAliasDoc, xml_alias_t::last_modified, xml_alias_t::name, and UPNP_E_OUTOF_MEMORY.

Referenced by UpnpUnRegisterRootDeviceLowPower().

9.93.3.25 web_server_set_cors()

```
int web_server_set_cors (
            const char * cors_string )
```

Assign the Access-Control-Allow-Origin specfied by the input const char* cors_string parameterto the global CORS string.

Returns

Integer.

Parameters

in	*cors_string*	String having the Access-Control-Allow-Origin string.

References gWebserverCorsString.

9.93.3.26 web_server_set_root_dir()

```
int web_server_set_root_dir (
            const char * root_dir )
```

Assign the path specfied by the input const char* root_dir parameter to the global Document root directory. Also check for path names ending in '/'.

Returns

Integer.

Parameters

in	*root_dir*	String having the root directory for the document.

References membuffer::buf, gDocumentRootDir, and membuffer::length.

9.93.4 Variable Documentation

9.93.4.1 gAliasDoc

```
struct xml_alias_t gAliasDoc  [static]
```

XML document.

Referenced by alias_grab(), glob_alias_init(), process_request(), web_server_destroy(), and web_server_set_alias().

9.93.4.2 gMediaTypeList

```
struct document_type_t gMediaTypeList[70]  [static]
```

module variables - Globals, static and externs.

Referenced by media_list_init(), and search_extension().

9.93.4.3 gMediaTypes

```
const char* gMediaTypes[]  [static]
```

Initial value:

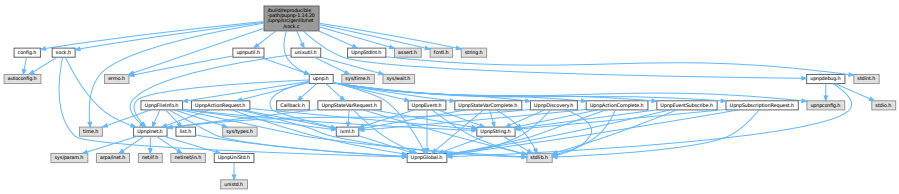
```
= {

        NULL,

        "audio",

        "video",

        "image",

        "application",

        "text"}
```

9.94 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/sock.c File Reference

Implements the sockets functionality.

```
#include "config.h"
#include "sock.h"
#include "UpnpStdInt.h"
#include "unixutil.h"
#include "upnp.h"
#include "upnpdebug.h"
#include "upnputil.h"
#include <assert.h>
#include <errno.h>
#include <fcntl.h>
#include <string.h>
#include <time.h>
```

Include dependency graph for sock.c:



Macros

- #define **MSG_NOSIGNAL** 0

Functions

- int sock_init (SOCKINFO *info, SOCKET sockfd)

  Assign the passed in socket descriptor to socket descriptor in the SOCKINFO structure.
- int sock_init_with_ip (SOCKINFO *info, SOCKET sockfd, struct sockaddr *foreign_sockaddr)

  Calls the sock_init function and assigns the passed in IP address and port to the IP address and port in the SOCKINFO structure.
- int sock_destroy (SOCKINFO *info, int ShutdownMethod)

  Associates an SSL object with the socket and begins the client-side SSL/TLS handshake.
- static int sock_read_write (SOCKINFO *info, char *buffer, size_t bufsize, int *timeoutSecs, int bRead)

  Receives or sends data. Also returns the time taken to receive or send data.
- int sock_read (SOCKINFO *info, char *buffer, size_t bufsize, int *timeoutSecs)

  Reads data on socket in sockinfo.
- int sock_write (SOCKINFO *info, const char *buffer, size_t bufsize, int *timeoutSecs)

  Writes data on the socket in sockinfo.
- int sock_make_blocking (SOCKET sock)

  Make socket blocking.
- int sock_make_no_blocking (SOCKET sock)

  Make socket non-blocking.

9.94.1 Detailed Description

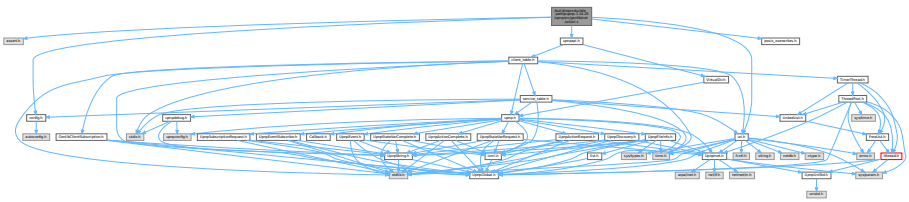
Implements the sockets functionality.

9.95 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/uri/uri.c File Reference

Contains functions for uri, url parsing utility.

```
#include <assert.h>
#include "config.h"
#include "upnpapi.h"
#include "uri.h"
#include "posix_overwrites.h"
```

Include dependency graph for uri.c:



Functions

- static int is_reserved (char in)

  Returns a 1 if a char is a RESERVED char as defined in http://www.ietf.org/rfc/rfc2396.txt RFC explaining URIs).
- int is_mark (char in)

  Returns a 1 if a char is a MARK char as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).
- int is_unreserved (char in)

  Returns a 1 if a char is an UNRESERVED char as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).
- int is_escaped (const char *in)

  Returns a 1 if a char[3] sequence is ESCAPED as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).
- int replace_escaped (char *in, size_t index, size_t *max)

  Replaces an escaped sequence with its unescaped version as in http://www.ietf.org/rfc/rfc2396.↩ txt (RFC explaining URIs)
- static size_t parse_uric (const char *in, size_t max, token *out)

  Parses a string of uric characters starting at in[0] as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).
- static void copy_token (const token *in, const char *in_base, token *out, char *out_base)

  Tokens are generally pointers into other strings. This copies the offset and size from a token (in) relative to one string (in_base) into a token (out) relative to another string (out_base).
- int copy_URL_list (URL_list *in, URL_list *out)

  Copies one URL_list into another.
- void free_URL_list (URL_list *list)

  Frees the memory associated with a URL_list.
- int token_string_casecmp (token *in1, const char *in2)

  Compares buffer in the token object with the buffer in in2.
- int token_cmp (token *in1, token *in2)

  Compares two tokens.
- static int parse_hostport (const char *in, unsigned short int defaultPort, hostport_type *out)

  Parses a string representing a host and port (e.g. "127.127.0.1:80" or "localhost") and fills out a hostport_type struct with internet address and a token representing the full host and port.
- static size_t parse_scheme (const char *in, size_t max, token *out)

  parses a uri scheme starting at in[0] as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).
- int remove_escaped_chars (char *in, size_t *size)

  Removes http escaped characters such as: "%20" and replaces them with their character representation. i.↩ e. "hello%20foo" -> "hello foo".
- static UPNP_INLINE int **is_end_path** (char c)
- int remove_dots (char *buf, size_t size)

  Removes ".", and ".." from a path.
- char * resolve_rel_url (char *base_url, char *rel_url)

  resolves a relative url with a base url returning a NEW (dynamically allocated with malloc) full url.
- int parse_uri (const char *in, size_t max, uri_type *out)

  Parses a uri as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

9.95.1 Detailed Description

Contains functions for uri, url parsing utility.

9.95.2 Function Documentation

9.95.2.1 copy_token()

```
static void copy_token (
            const token * in,
            const char * in_base,
            token * out,
            char * out_base )  [static]
```

Tokens are generally pointers into other strings. This copies the offset and size from a token (in) relative to one string (in_base) into a token (out) relative to another string (out_base).

Parameters

in	*in*	Source token.
in	*in_base*	.
out	*out*	Destination token.
in	*out_base*	.

Referenced by copy_URL_list().

9.95.2.2 copy_URL_list()

```
int copy_URL_list (
            URL_list * in,
            URL_list * out )
```

Copies one URL_list into another.

This includes dynamically allocating the out->URLs field (the full string), and the structures used to hold the parsedURLs. This memory MUST be freed by the caller through: free_URL_list(&out).

Returns

- HTTP_SUCCESS - On Success.
- UPNP_E_OUTOF_MEMORY - On Failure to allocate memory.

Parameters

in	*in*	Source URL list.
out	*out*	Destination URL list.

References copy_token(), HOSTPORT::text, UPNP_E_OUTOF_MEMORY, and URL_LIST::URLs.

Referenced by copy_subscription().

9.95.2.3 free_URL_list()

```
void free_URL_list (
            URL_list * list )
```

Frees the memory associated with a URL_list.

Frees the dynamically allocated members of of list. Does NOT free the pointer to the list itself (i.e. does NOT free(list)).

Parameters

in	*list*	URL list object.

References URL_LIST::URLs.

Referenced by freeSubscription().

9.95.2.4 is_escaped()

```
int is_escaped (
            const char * in )
```

Returns a 1 if a char[3] sequence is ESCAPED as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

Size of array is NOT checked (MUST be checked by caller).

Returns

1 if char is a ESCAPED char.

Parameters

in	*in*	Char sequence to be matched for ESCAPED characters.

Referenced by parse_uric().

9.95.2.5 is_mark()

```
int is_mark (
            char in )
```

Returns a 1 if a char is a MARK char as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

Returns

1 if char is a MARKED char.

Parameters

in	*in*	Char to be matched for MARKED characters.

Referenced by is_unreserved().

9.95.2.6 is_reserved()

```
static int is_reserved (
            char in )  [static]
```

Returns a 1 if a char is a RESERVED char as defined in http://www.ietf.org/rfc/rfc2396.txt RFC explaining URIs).

Returns

1 if char is a RESERVED char.

Parameters

in	*in*	Char to be matched for RESERVED characters.

References RESERVED.

Referenced by parse_uric().

9.95.2.7 is_unreserved()

```
int is_unreserved (
            char in )
```

Returns a 1 if a char is an UNRESERVED char as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

Returns

1 if char is a UNRESERVED char.

Parameters

in	*in*	Char to be matched for UNRESERVED characters.

References is_mark().

Referenced by parse_uric().

9.95.2.8 parse_hostport()

```
static int parse_hostport (
            const char * in,
            unsigned short int defaultPort,
            hostport_type * out )  [static]
```

Parses a string representing a host and port (e.g. "127.127.0.1:80" or "localhost") and fills out a hostport_type struct with internet address and a token representing the full host and port.

Parameters

in	*in*	String of characters representing host and port.
out	*defaultPort*	Output parameter where the host and port are represented as an internet address.

References gIF_INDEX, HOSTPORT::text, and UPNP_E_INVALID_URL.

Referenced by parse_uri().

9.95.2.9 parse_scheme()

```
static size_t parse_scheme (
            const char * in,
            size_t max,
            token * out )  [static]
```

parses a uri scheme starting at in[0] as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

(e.g. "http:" -> scheme= "http").

Note

String MUST include ':' within the max charcters.

Returns

Parameters

in	*in*	String of characters representing a scheme.
in	*max*	Maximum number of characters.
out	*out*	Output parameter whose buffer is filled in with the scheme.

Referenced by parse_uri().

9.95.2.10 parse_uri()

```
int parse_uri (
            const char * in,
            size_t max,
            uri_type * out )
```

Parses a uri as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

Handles absolute, relative, and opaque uris. Parses into the following pieces: scheme, hostport, pathquery, fragment (path and query are treated as one token)

Caller should check for the pieces they require.

Returns

Parameters

in	*in*	Character string containing uri information to be parsed.
in	*max*	Maximum limit on the number of characters.
out	*out*	Output parameter which will have the parsed uri information.

References parse_hostport(), parse_scheme(), parse_uric(), and token_string_casecmp().

Referenced by create_url_list(), FindServiceControlURLPath(), FindServiceEventURLPath(), http_FixStrUrl(), and resolve_rel_url().

9.95.2.11 parse_uric()

```
static size_t parse_uric (
            const char * in,
            size_t max,
            token * out )  [static]
```

Parses a string of uric characters starting at in[0] as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

Returns

Parameters

in	*in*	String of characters.
in	*max*	Maximum limit.
out	*out*	Token object where the string of characters is copied.

References is_escaped(), is_reserved(), and is_unreserved().

Referenced by parse_uri().

9.95.2.12 remove_dots()

```
int remove_dots (
            char * in,
            size_t size )
```

Removes ".", and ".." from a path.

If a ".." can not be resolved (i.e. the .. would go past the root of the path) an error is returned.

The input IS modified in place.)

Note

Examples char path[30]="/../hello"; remove_dots(path, strlen(path)) -> UPNP_E_INVALID_URL char path[30]="/./hello"; remove_dots(path, strlen(path)) -> UPNP_E_SUCCESS, in = "/hello" char path[30]="/./hello/foo/../goodbye" -> UPNP_E_SUCCESS, in = "/hello/goodbye"

Returns

- UPNP_E_SUCCESS - On Success.
- UPNP_E_OUTOF_MEMORY - On failure to allocate memory.
- UPNP_E_INVALID_URL - Failure to resolve URL.

Parameters

in	*buf*	String of characters from which "dots" have to be removed.
in	*size*	Size limit for the number of characters.

References UPNP_E_SUCCESS.

Referenced by process_request(), and resolve_rel_url().

9.95.2.13 remove_escaped_chars()

```
int remove_escaped_chars (
            char * in,
            size_t * size )
```

Removes http escaped characters such as: "%20" and replaces them with their character representation. i.e. "hello%20foo" -> "hello foo".

The input IS MODIFIED in place (shortened). Extra characters are replaced with **NULL**.

Returns

UPNP_E_SUCCESS.

Parameters

in,out	*in*	String of characters to be modified.
in,out	*size*	Size limit for the number of characters.

References replace_escaped(), and UPNP_E_SUCCESS.

Referenced by process_request().

9.95.2.14 replace_escaped()

```
int replace_escaped (
            char * in,
```

```
            size_t index,
            size_t * max )
```

Replaces an escaped sequence with its unescaped version as in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs)

Size of array is NOT checked (MUST be checked by caller)

Note

This function modifies the string. If the sequence is an escaped sequence it is replaced, the other characters in the string are shifted over, and NULL characters are placed at the end of the string.

Returns

Parameters

in,out	*in*	String of characters.
in	*index*	Index at which to start checking the characters.
out	*max*	.

Referenced by remove_escaped_chars().

9.95.2.15 resolve_rel_url()

```
char * resolve_rel_url (
            char * base_url,
            char * rel_url )
```

resolves a relative url with a base url returning a NEW (dynamically allocated with malloc) full url.

If the base_url is **NULL**, then a copy of the rel_url is passed back if the rel_url is absolute then a copy of the rel_url is passed back if neither the base nor the rel_url are Absolute then NULL is returned. Otherwise it tries and resolves the relative url with the base as described in http://www.ietf.org/rfc/rfc2396.txt (RFCs explaining URIs).

The resolution of '..' is NOT implemented, but '.' is resolved.

Returns

Parameters

in	*base_url*	Base URL.
in	*rel_url*	Relative URL.

References parse_uri(), remove_dots(), HOSTPORT::text, and UPNP_E_SUCCESS.

Referenced by UpnpResolveURL(), and UpnpResolveURL2().

9.95.2.16 token_cmp()

```
int token_cmp (
            token * in1,
            token * in2 )
```

Compares two tokens.

Returns

- < 0, if string1 is less than string2.
- == 0, if string1 is identical to string2 .
- > 0, if string1 is greater than string2.

Parameters

in	*in1*	First token object whose buffer is to be compared.
in	*in2*	Second token object used for the comparison.

Referenced by FindServiceControlURLPath(), and FindServiceEventURLPath().

9.95.2.17 token_string_casecmp()

```
int token_string_casecmp (
            token * in1,
            const char * in2 )
```

Compares buffer in the token object with the buffer in in2.

Returns

- < 0, if string1 is less than string2.
- == 0, if string1 is identical to string2 .
- > 0, if string1 is greater than string2.

Parameters

in	*in1*	Token object whose buffer is to be compared.
in	*in2*	String of characters to compare with.

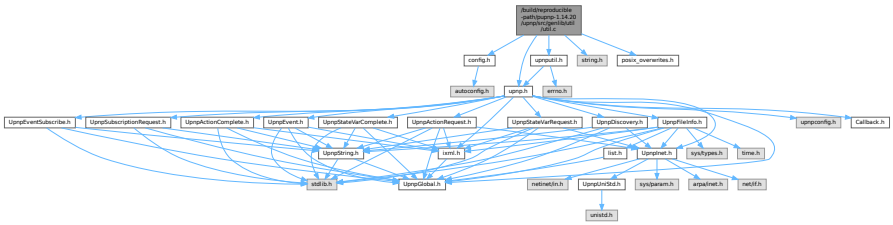
Referenced by http_FixUrl(), http_OpenHttpConnection(), and parse_uri().

9.96 /build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/util/util.c File Reference

```
#include "config.h"
#include "upnp.h"
```

```
#include "upnputil.h"
#include <string.h>
#include "posix_overwrites.h"
```

Include dependency graph for util.c:



Functions

- void linecopy (char dest[LINE_SIZE], const char *src)

  Copy no of bytes spcified by the LINE_SIZE constant, from the source buffer. Null terminate the destination buffer.
- void namecopy (char dest[NAME_SIZE], const char *src)

  Copy no of bytes spcified by the NAME_SIZE constant, from the source buffer. Null terminate the destination buffer.
- void linecopylen (char dest[LINE_SIZE], const char *src, size_t srclen)

  Determine if the srclen passed in paramter is less than the permitted LINE_SIZE. If it is use the passed parameter, if not use the permitted LINE_SIZE as the length parameter.

9.96.1 Detailed Description

Purpose: This file contains functions for copying strings based on different options.

9.96.2 Function Documentation

9.96.2.1 linecopy()

```
void linecopy (
            char dest[LINE_SIZE],
            const char * src )
```

Copy no of bytes spcified by the LINE_SIZE constant, from the source buffer. Null terminate the destination buffer.

Parameters

out	*dest*	output buffer.
in	*src*	input buffer.

9.96.2.2 linecopylen()

```
void linecopylen (
            char dest[LINE_SIZE],
```

```
            const char * src,
            size_t srclen )
```

Determine if the srclen passed in paramter is less than the permitted LINE_SIZE. If it is use the passed parameter, if not use the permitted LINE_SIZE as the length parameter.

Copy no of bytes spcified by the LINE_SIZE constant, from the source buffer. Null terminate the destination buffer.

Parameters

out	*dest*	output buffer.
in	*src*	input buffer.
in	*srclen*	bytes to be copied.

9.96.2.3 namecopy()

```
void namecopy (
            char dest[NAME_SIZE],
            const char * src )
```

Copy no of bytes spcified by the NAME_SIZE constant, from the source buffer. Null terminate the destination buffer.

Parameters

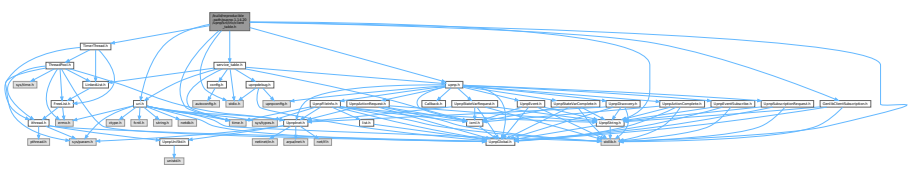
out	*dest*	output buffer.
in	*src*	input buffer.

Referenced by check_soapaction_hdr(), and get_dev_service().

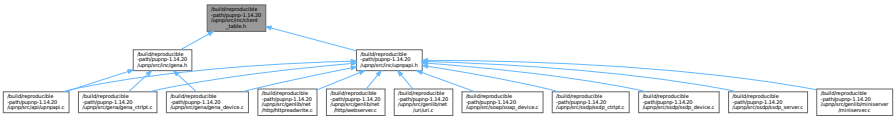
9.97 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/client_table.h File Reference

```
#include "GenlibClientSubscription.h"
#include "TimerThread.h"
#include "UpnpString.h"
#include "service_table.h"
#include "upnp.h"
#include "uri.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for client_table.h:



This graph shows which files directly or indirectly include this file:



Functions

- void free_client_subscription (GenlibClientSubscription *sub)

  Free memory allocated for client subscription data.
- void freeClientSubList (GenlibClientSubscription *list)

  Free the client subscription table.
- void RemoveClientSubClientSID (GenlibClientSubscription **head, const UpnpString *sid)

  Remove the client subscription matching the subscritpion id represented by the const Upnp_SID sid parameter from the table and update the table.
- GenlibClientSubscription * GetClientSubClientSID (GenlibClientSubscription *head, const UpnpString *sid)

  Return the client subscription from the client table that matches const Upnp_SID sid subscrition id value.
- GenlibClientSubscription * GetClientSubActualSID (GenlibClientSubscription *head, token *sid)

  *Returns the client subscription from the client subscription table that has the matching token *sid buffer value.*

Variables

- TimerThread gTimerThread

9.97.1 Function Documentation

9.97.1.1 free_client_subscription()

```
void free_client_subscription (
            GenlibClientSubscription * sub )
```

Free memory allocated for client subscription data.

Remove timer thread associated with this subscription event.

Parameters

in	*sub*	Client subscription to be freed.

References GenlibClientSubscription_get_RenewEventId(), GenlibClientSubscription_set_RenewEventId(), GenlibClientSubscription_strcpy_ActualSID(), GenlibClientSubscription_strcpy_EventURL(), gTimerThread, and TimerThreadRemove().

9.97.1.2 freeClientSubList()

```
void freeClientSubList (
            GenlibClientSubscription * list )
```

Free the client subscription table.

Parameters

in	*list*	Client subscription list to be freed.

References GenlibClientSubscription_delete(), and GenlibClientSubscription_get_Next().

9.97.1.3 GetClientSubActualSID()

```
GenlibClientSubscription * GetClientSubActualSID (
            GenlibClientSubscription * head,
            token * sid )
```

Returns the client subscription from the client subscription table that has the matching token *sid buffer value.

Returns

The matching subscription.

Parameters

in	*head*	Head of the subscription list.
in	*sid*	Subscription ID to be mactched.

References GenlibClientSubscription_get_ActualSID_cstr(), and GenlibClientSubscription_get_Next().

9.97.1.4 GetClientSubClientSID()

```
GenlibClientSubscription * GetClientSubClientSID (
            GenlibClientSubscription * head,
            const UpnpString * sid )
```

Return the client subscription from the client table that matches const Upnp_SID sid subscrition id value.

Returns

The matching subscription.

Parameters

in	*head*	Head of the subscription list.
in	*sid*	Subscription ID to be mactched.

References GenlibClientSubscription_get_Next(), GenlibClientSubscription_get_SID_cstr(), and UpnpString_get_String().

9.97.1.5 RemoveClientSubClientSID()

```
void RemoveClientSubClientSID (
            GenlibClientSubscription ** head,
            const UpnpString * sid )
```

Remove the client subscription matching the subscritpion id represented by the const Upnp_SID sid parameter from the table and update the table.

Parameters

in	*head*	Head of the subscription list.
in	*sid*	Subscription ID to be mactched.

References GenlibClientSubscription_get_Next(), GenlibClientSubscription_get_SID_cstr(), GenlibClientSubscription_set_Next(), and UpnpString_get_String().

9.98 client_table.h

Go to the documentation of this file.

```
00001
00002 #ifndef CLIENT_TABLE_H
00003 #define CLIENT_TABLE_H
00004
00009 #ifdef __cplusplus
00010 extern "C" {
00011 #endif
00012
00013 #include "GenlibClientSubscription.h"
00014 #include "TimerThread.h"
00015 #include "UpnpString.h"
00016 #include "service_table.h"
00017 #include "upnp.h"
00018 #include "uri.h"
00019
00020 #include <stdio.h>
00021 #include <stdlib.h>
00022 #include <time.h>
00023
00024 extern TimerThread gTimerThread;
00025
00026 #ifdef INCLUDE_CLIENT_APIS
00027
00033 void free_client_subscription(
00035         GenlibClientSubscription *sub);
00036
00040 void freeClientSubList(
00042         GenlibClientSubscription *list);
00043
00049 void RemoveClientSubClientSID(
00051         GenlibClientSubscription **head,
00053         const UpnpString *sid);
00054
00061 GenlibClientSubscription *GetClientSubClientSID(
00063         GenlibClientSubscription *head,
00065         const UpnpString *sid);
00066
00073 GenlibClientSubscription *GetClientSubActualSID(
00075         GenlibClientSubscription *head,
00077         token *sid);
00078
00079 #endif /* INCLUDE_CLIENT_APIS */
00080
00081 #ifdef __cplusplus
00082 }
00083 #endif /* __cplusplus */
00084
00085 #endif /* CLIENT_TABLE_H */
```

9.99 config.h

```
00001 /*****************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (c) 2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * - Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * - Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * - Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  *****************************************************************************/
00032
00033 #ifndef INTERNAL_CONFIG_H
00034 #define INTERNAL_CONFIG_H
00035
00036 #include "autoconfig.h"
00037
00059 #define THREAD_IDLE_TIME 5000
00060 /* @} */
00061
00073 #define JOBS_PER_THREAD 10
00074 /* @} */
00075
00088 #define MIN_THREADS 2
00089 /* @} */
00090
00105 #define MAX_THREADS 12
00106 /* @} */
00107
00122 #define THREAD_STACK_SIZE (size_t)0
00123 /* @} */
00124
00134 #define MAX_JOBS_TOTAL 100
00135 /* @} */
00136
00146 #define MAX_SUBSCRIPTION_QUEUED_EVENTS 10
00147 /* @} */
00148
00162 #define MAX_SUBSCRIPTION_EVENT_AGE 30
00163 /* @} */
00164
00175 #define DEFAULT_SOAP_CONTENT_LENGTH 16000
00176 /* @} */
00177
00187 #define NUM_SSDP_COPY 2
00188 /* @} */
00189
00199 #define SSDP_PAUSE 100u
00200 /* @} */
00201
00210 #define WEB_SERVER_BUF_SIZE (size_t)(1024 * 1024)
00211 /* @} */
00212
00224 #define WEB_SERVER_CONTENT_LANGUAGE ""
00225 /* @} */
00226
00240 #define AUTO_RENEW_TIME 10
00241 /* @} */
00242
00254 #define CP_MINIMUM_SUBSCRIPTION_TIME (AUTO_RENEW_TIME + 5)
00255 /* @} */
00256
00267 #define MAX_SEARCH_TIME 80
00268 /* @} */
00269
```

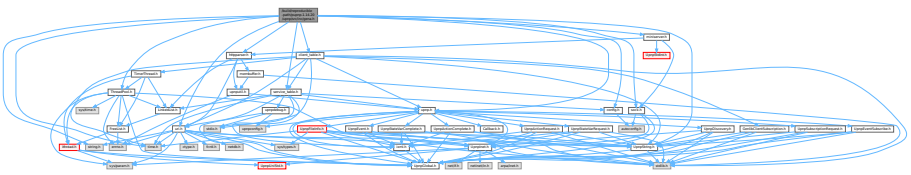
```
00280 #define MIN_SEARCH_TIME 2
00281 /* @} */
00282
00292 #define AUTO_ADVERTISEMENT_TIME 30
00293 /* @} */
00294
00307 #define SSDP_PACKET_DISTRIBUTE 1
00308 /* @} */
00309
00327 #define GENA_NOTIFICATION_SENDING_TIMEOUT HTTP_DEFAULT_TIMEOUT
00328 /* @} */
00329
00349 #define GENA_NOTIFICATION_ANSWERING_TIMEOUT HTTP_DEFAULT_TIMEOUT
00350 /* @} */
00351
00372 #define EXCLUDE_SSDP 0
00373 #define EXCLUDE_SOAP 0
00374 #define EXCLUDE_GENA 0
00375 #define EXCLUDE_DOM 0
00376 #define EXCLUDE_MINISERVER 0
00377 #define EXCLUDE_WEB_SERVER 0
00378 #ifdef USE_JNI
00379         #define EXCLUDE_JNI 0
00380 #else
00381         #define EXCLUDE_JNI 1
00382 #endif
00383 /* @} */
00384
00392 #define DEBUG_ALL 1
00393 #define DEBUG_SSDP 0
00394 #define DEBUG_SOAP 0
00395 #define DEBUG_GENA 0
00396 #define DEBUG_TPOOL 0
00397 #define DEBUG_MSERV 0
00398 #define DEBUG_DOM 0
00399 #define DEBUG_HTTP 0
00400 #define DEBUG_API 0
00401
00402 /*
00403  * @} Compile time configuration options
00404  */
00405
00406 /*****************************************************************************
00407  * Do not change, Internal purpose only!!!
00408  ****************************************************************************/
00409
00414 /*
00415  * Set additional defines based on requested configuration
00416  */
00417
00418 /* configure --enable-client */
00419 #if UPNP_HAVE_CLIENT
00420         #define INCLUDE_CLIENT_APIS 1
00421 #endif
00422
00423 /* configure --enable-device */
00424 #if UPNP_HAVE_DEVICE
00425         #define INCLUDE_DEVICE_APIS 1
00426 #endif
00427
00428 /* configure --enable-webserver */
00429 #if UPNP_HAVE_WEBSERVER
00430         #define INTERNAL_WEB_SERVER 1
00431 #endif
00432
00433 /* configure --enable-ssdp */
00434 #undef EXCLUDE_SSDP
00435 #if UPNP_HAVE_SSDP
00436         #define EXCLUDE_SSDP 0
00437 #else
00438         #define EXCLUDE_SSDP 1
00439 #endif
00440
00441 /* configure --enable-soap */
00442 #undef EXCLUDE_SOAP
00443 #if UPNP_HAVE_SOAP
00444         #define EXCLUDE_SOAP 0
00445 #else
00446         #define EXCLUDE_SOAP 1
00447 #endif
00448
00449 /* configure --enable-gena */
00450 #undef EXCLUDE_GENA
00451 #if UPNP_HAVE_GENA
00452         #define EXCLUDE_GENA 0
00453 #else
00454         #define EXCLUDE_GENA 1
```

```
00455 #endif
00456
00457 #undef EXCLUDE_WEB_SERVER
00458 #undef EXCLUDE_MINISERVER
00459 #ifdef INTERNAL_WEB_SERVER
00460         #define EXCLUDE_WEB_SERVER 0
00461         #define EXCLUDE_MINISERVER 0
00462 #else
00463         #define EXCLUDE_WEB_SERVER 1
00464         #define EXCLUDE_MINISERVER 1
00465 #endif
00466
00467 #if EXCLUDE_SSDP == 1 && EXCLUDE_GENA == 1 && EXCLUDE_SOAP == 1 && \
00468         EXCLUDE_WEB_SERVER == 1
00469         #undef EXCLUDE_MINISERVER
00470         #define EXCLUDE_MINISERVER 1
00471         #if INTERNAL_WEB_SERVER
00472                 #error "conflicting settings: use configure --disable-webserver"
00473         #endif
00474 #endif
00475
00476 #if EXCLUDE_SSDP == 0 || EXCLUDE_GENA == 0 || EXCLUDE_SOAP == 0 || \
00477         EXCLUDE_WEB_SERVER == 0
00478         #undef EXCLUDE_MINISERVER
00479         #define EXCLUDE_MINISERVER 0
00480         #if EXCLUDE_WEB_SERVER == 0 && !defined INTERNAL_WEB_SERVER
00481                 #error "conflicting settings : use configure --enable-webserver"
00482         #endif
00483 #endif
00484
00485 /*
00486  * @}
00487  */
00488
00489 #endif /* INTERNAL_CONFIG_H */
```

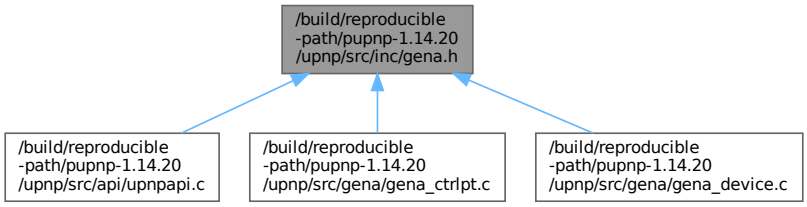
9.100 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena.h File Reference

```
#include "config.h"
#include <string.h>
#include <time.h>
#include "ThreadPool.h"
#include "UpnpString.h"
#include "client_table.h"
#include "httpparser.h"
#include "miniserver.h"
#include "service_table.h"
#include "sock.h"
#include "upnp.h"
#include "uri.h"
```

Include dependency graph for gena.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct NOTIFY_THREAD_STRUCT

Macros

- #define **EXTERN_C**
- #define XML_VERSION "<?xml version='1.0' encoding='ISO-8859-1' ?>\n"

  XML version comment. Not used because it is not interopeable with other UPnP vendors.
- #define **XML_PROPERTYSET_HEADER** "<e:propertyset xmlns:e=\"urn:schemas-upnp-org:event-1-0\">\n"
- #define **UNABLE_MEMORY** "HTTP/1.1 500 Internal Server Error\r\n\r\n"
- #define **UNABLE_SERVICE_UNKNOWN** "HTTP/1.1 404 Not Found\r\n\r\n"
- #define **UNABLE_SERVICE_NOT_ACCEPT** "HTTP/1.1 503 Service Not Available\r\n\r\n"
- #define **NOT_IMPLEMENTED** "HTTP/1.1 501 Not Implemented\r\n\r\n"
- #define **BAD_REQUEST** "HTTP/1.1 400 Bad Request\r\n\r\n"
- #define **INVALID_NT** BAD_CALLBACK
- #define **BAD_CALLBACK** "HTTP/1.1 412 Precondition Failed\r\n\r\n"
- #define **HTTP_OK_CRLF** "HTTP/1.1 200 OK\r\n\r\n"
- #define **HTTP_OK_STR** "HTTP/1.1 200 OK\r\n"
- #define **INVALID_SID** BAD_CALLBACK
- #define **MISSING_SID** BAD_CALLBACK
- #define **MAX_CONTENT_LENGTH** 20
- #define **MAX_SECONDS** 10
- #define **MAX_EVENTS** 20
- #define **MAX_PORT_SIZE** 10
- #define **GENA_E_BAD_RESPONSE** UPNP_E_BAD_RESPONSE
- #define **GENA_E_BAD_SERVICE** UPNP_E_INVALID_SERVICE
- #define **GENA_E_SUBSCRIPTION_UNACCEPTED** UPNP_E_SUBSCRIBE_UNACCEPTED
- #define **GENA_E_BAD_SID** UPNP_E_INVALID_SID
- #define **GENA_E_UNSUBSCRIBE_UNACCEPTED** UPNP_E_UNSUBSCRIBE_UNACCEPTED
- #define **GENA_E_NOTIFY_UNACCEPTED** UPNP_E_NOTIFY_UNACCEPTED
- #define **GENA_E_NOTIFY_UNACCEPTED_REMOVE_SUB** -9
- #define **GENA_E_BAD_HANDLE** UPNP_E_INVALID_HANDLE
- #define **XML_ERROR** -5
- #define **XML_SUCCESS** UPNP_E_SUCCESS
- #define **GENA_SUCCESS** UPNP_E_SUCCESS
- #define **CALLBACK_SUCCESS** 0
- #define **DEFAULT_TIMEOUT** 1801
- #define SubscribeLock()

  Locks the subscription.
- #define SubscribeUnlock()

  Unlocks the subscription.

Typedefs

- typedef struct NOTIFY_THREAD_STRUCT notify_thread_struct

Functions

- void genaCallback (http_parser_t *parser, http_message_t *request, SOCKINFO *info)

  This is the callback function called by the miniserver to handle incoming GENA requests.
- int genaSubscribe (UpnpClient_Handle client_handle, const UpnpString *PublisherURL, int *TimeOut, UpnpString *out_sid)

  This function subscribes to a PublisherURL (also mentioned as EventURL in some places).
- int genaUnSubscribe (UpnpClient_Handle client_handle, const UpnpString *in_sid)

  Unsubscribes a SID.
- int genaUnregisterClient (UpnpClient_Handle client_handle)

  Unsubcribes all the outstanding subscriptions and cleans the subscription list.
- int genaUnregisterDevice (UpnpDevice_Handle device_handle)

  Cleans the service table of the device.
- int genaRenewSubscription (UpnpClient_Handle client_handle, const UpnpString *in_sid, int *TimeOut)

  Renews a SID.
- int genaNotifyAll (UpnpDevice_Handle device_handle, char *UDN, char *servId, char **VarNames, char **VarValues, int var_count)

  Sends a notification to all the subscribed control points.
- int genaNotifyAllExt (UpnpDevice_Handle device_handle, char *UDN, char *servId, IXML_Document *PropSet)

  Sends a notification to all the subscribed control points.
- int genaInitNotify (UpnpDevice_Handle device_handle, char *UDN, char *servId, char **VarNames, char **VarValues, int var_count, const Upnp_SID sid)

  Sends the intial state table dump to newly subscribed control point.
- int genaInitNotifyExt (UpnpDevice_Handle device_handle, char *UDN, char *servId, IXML_Document *PropSet, const Upnp_SID sid)

  Similar to the genaInitNofity. The only difference is that it takes the xml document for the state table and sends the intial state table dump to newly subscribed control point.
- void error_respond (SOCKINFO *info, int error_code, http_message_t *hmsg)

  Sends an error message to the control point in the case of incorrect GENA requests.

Variables

- ithread_mutex_t GlobalClientSubscribeMutex

9.100.1 Macro Definition Documentation

9.100.1.1 SubscribeLock

```
#define SubscribeLock( )
```

Value:

```
UpnpPrintf( \
        UPNP_INFO, GENA, __FILE__, __LINE__, "Trying Subscribe Lock"); \
ithread_mutex_lock(&GlobalClientSubscribeMutex); \
UpnpPrintf(UPNP_INFO, GENA, __FILE__, __LINE__, "Subscribe Lock");
```

Locks the subscription.

9.100.1.2 SubscribeUnlock

```
#define SubscribeUnlock( )
```

Value:

```
        UpnpPrintf(UPNP_INFO, \
                GENA, \
                __FILE__, \
                __LINE__, \
                "Trying Subscribe UnLock"); \
        ithread_mutex_unlock(&GlobalClientSubscribeMutex); \
        UpnpPrintf(UPNP_INFO, GENA, __FILE__, __LINE__, "Subscribe UnLock");
```

Unlocks the subscription.

9.100.1.3 XML_VERSION

```
#define XML_VERSION "<?xml version='1.0' encoding='ISO-8859-1' ?>\n"
```

XML version comment. Not used because it is not interopeable with other UPnP vendors.

9.100.2 Typedef Documentation

9.100.2.1 notify_thread_struct

```
typedef struct NOTIFY_THREAD_STRUCT notify_thread_struct
```

Structure to send NOTIFY message to all subscribed control points

9.100.3 Function Documentation

9.100.3.1 error_respond()

```
void error_respond (
             SOCKINFO * info,
             int error_code,
             http_message_t * hmsg )
```

Sends an error message to the control point in the case of incorrect GENA requests.

Returns

UPNP_E_SUCCESS if successful, otherwise appropriate error code.

Parameters

in	*info*	Structure containing information about the socket.
in	*error_code*	error code that will be in the GENA response.
in	*hmsg*	GENA request Packet.

9.100.3.2 genaCallback()

```
void genaCallback (
            http_parser_t * parser,
            http_message_t * request,
            SOCKINFO * info )
```

This is the callback function called by the miniserver to handle incoming GENA requests.

Returns

UPNP_E_SUCCESS if successful, otherwise appropriate error code.

Parameters

in	*parser*	represents the parse state of the request
in	*request*	HTTP message containing GENA request
in,out	*info*	Structure containing information about the socket

References gena_process_notification_event(), gena_process_subscription_renewal_request(), gena_process_subscription_request(), gena_process_unsubscribe_request(), http_message_t::method, and UpnpPrintf().

9.100.3.3 genaInitNotify()

```
int genaInitNotify (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            char ** VarNames,
            char ** VarValues,
            int var_count,
            const Upnp_SID sid )
```

Sends the intial state table dump to newly subscribed control point.

Returns

GENA_E_SUCCESS if successful, otherwise the appropriate error code.

Note

No other event will be sent to this control point before the intial state table dump.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*VarNames*	Array of variable names.
in	*VarValues*	Array of variable values.
in	*var_count*	Array size.
in	*sid*	Subscription ID.

References device_handle, DOMString, GeneratePropertySet(), and UpnpPrintf().

Referenced by UpnpAcceptSubscription().

9.100.3.4 genaInitNotifyExt()

```
int genaInitNotifyExt (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            IXML_Document * PropSet,
            const Upnp_SID sid )
```

Similar to the genaInitNofity. The only difference is that it takes the xml document for the state table and sends the intial state table dump to newly subscribed control point.

Returns

GENA_E_SUCCESS if successful, otherwise the appropriate error code.

Note

No other event will be sent to this control point before the intial state table dump.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*PropSet*	Document of the state table.
in	*sid*	subscription ID.

References device_handle, DOMString, ixmlPrintNode(), UPNP_E_INVALID_PARAM, and UpnpPrintf().

Referenced by UpnpAcceptSubscriptionExt().

9.100.3.5 genaNotifyAll()

```
int genaNotifyAll (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            char ** VarNames,
            char ** VarValues,
            int var_count )
```

Sends a notification to all the subscribed control points.

Returns

int

Note

This function is similar to the genaNotifyAllExt. The only difference is it takes event variable array instead of xml document.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*VarNames*	Array of varible names.
in	*VarValues*	Array of variable values.
in	*var_count*	Number of variables.

References device_handle, DOMString, GeneratePropertySet(), and UpnpPrintf().

Referenced by UpnpNotify().

9.100.3.6 genaNotifyAllExt()

```
int genaNotifyAllExt (
            UpnpDevice_Handle device_handle,
            char * UDN,
            char * servId,
            IXML_Document * PropSet )
```

Sends a notification to all the subscribed control points.

Returns

int

Note

This function is similar to the genaNotifyAll. the only difference is it takes the document instead of event variable array.

Parameters

in	*device_handle*	Device handle.
in	*UDN*	Device udn.
in	*servId*	Service ID.
in	*PropSet*	XML document Event varible property set.

References device_handle, DOMString, ixmlPrintNode(), UPNP_E_INVALID_PARAM, and UpnpPrintf().

Referenced by UpnpNotifyExt().

9.100.3.7 genaRenewSubscription()

```
int genaRenewSubscription (
```

```
            UpnpClient_Handle client_handle,
            const UpnpString * in_sid,
            int * TimeOut )
```

Renews a SID.

It first validates the SID and client_handle and copies the subscription. It sends RENEW (modified SUBSCRIBE) http request to service and processes the response.

Returns

UPNP_E_SUCCESS if service response is OK, otherwise the appropriate error code.

Parameters

in	*client_handle*	Client handle.
in	*in_sid*	Subscription ID.
in,out	*TimeOut*	requested Duration, if -1, then "infinite". In the OUT case: actual Duration granted by Service, -1 for infinite.

References Handle_Info::ClientSubList, gena_subscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_delete(), GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_get_RenewEventId(GenlibClientSubscription_get_SID(), GenlibClientSubscription_new(), GenlibClientSubscription_set_ActualSID(), GenlibClientSubscription_set_RenewEventId(), GetHandleInfo(), gTimerThread, ScheduleGenaAutoRenew(), TimerThreadRemove(), UPNP_E_SUCCESS, UpnpPrintf(), UpnpString_delete(), and UpnpString_new().

Referenced by GenaAutoRenewSubscription(), UpnpRenewSubscription(), and UpnpThreadDistribution().

9.100.3.8 genaSubscribe()

```
int genaSubscribe (
            UpnpClient_Handle client_handle,
            const UpnpString * PublisherURL,
            int * TimeOut,
            UpnpString * out_sid )
```

This function subscribes to a PublisherURL (also mentioned as EventURL in some places).

It sends SUBSCRIBE http request to service processes request. Finally adds a Subscription to the clients subscription list, if service responds with OK.

Returns

UPNP_E_SUCCESS if service response is OK, otherwise returns the appropriate error code

Parameters

in	*client_handle*	The client handle.
in	*PublisherURL*	Of the form: " http://134.134.156.80:4000/RedBulb/Event
in,out	*TimeOut*	requested Duration: • if -1, then "infinite". • in the OUT case: actual Duration granted by Service, -1 for infinite.
out	*out_sid*	sid of subscription, memory passed in by caller.

References Handle_Info::ClientSubList, gena_subscribe(), GenlibClientSubscription_delete(), GenlibClientSubscription_new(), GenlibClientSubscription_set_ActualSID(), GenlibClientSubscription_set_EventURL(), GenlibClientSubscription_set_Next(), GenlibClientSubscription_set_RenewEventId(), GenlibClientSubscription_set_SID(), GetHandleInfo(), ScheduleGenaAutoRenew(), SubscribeLock, SubscribeUnlock, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UpnpPrintf(), UpnpString_assign(), UpnpString_clear(), UpnpString_delete(), UpnpString_new(), and UpnpString_set_String().

Referenced by UpnpSubscribe(), and UpnpThreadDistribution().

9.100.3.9 genaUnregisterClient()

```
int genaUnregisterClient (
            UpnpClient_Handle client_handle )
```

Unsubcribes all the outstanding subscriptions and cleans the subscription list.

This function is called when control point unregisters.

Returns

UPNP_E_SUCCESS if successful, otherwise returns the appropriate error code.

Parameters

in	*client_handle*	Handle containing all the control point related information.

References Handle_Info::ClientSubList, gena_unsubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_delete(), GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_get_SID(), GenlibClientSubscription_new(), GetHandleInfo(), and UPNP_E_SUCCESS.

Referenced by UpnpUnRegisterClient().

9.100.3.10 genaUnregisterDevice()

```
int genaUnregisterDevice (
            UpnpDevice_Handle device_handle )
```

Cleans the service table of the device.

Returns

UPNP_E_SUCCESS if successful, otherwise returns GENA_E_BAD_HANDLE

Cleans the service table of the device.

Returns

UPNP_E_SUCCESS on success, GENA_E_BAD_HANDLE on failure.

Parameters

in	*device_handle*	Device handle.

References device_handle, GetHandleInfo(), Handle_Info::ServiceTable, UPNP_E_SUCCESS, and UpnpPrintf().

Referenced by UpnpUnRegisterRootDeviceLowPower().

9.100.3.11 genaUnSubscribe()

```
int genaUnSubscribe (
            UpnpClient_Handle client_handle,
            const UpnpString * in_sid )
```

Unsubscribes a SID.

It first validates the SID and client_handle,copies the subscription, sends UNSUBSCRIBE http request to service processes request and finally removes the subscription.

Returns

UPNP_E_SUCCESS if service response is OK, otherwise returns the appropriate error code.

Parameters

in	*client_handle*	UPnP client handle.
in	*in_sid*	The subscription ID.

References Handle_Info::ClientSubList, gena_unsubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_delete(), GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_new(), and GetHandleInfo().

Referenced by UpnpThreadDistribution(), and UpnpUnSubscribe().

9.101 gena.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
```

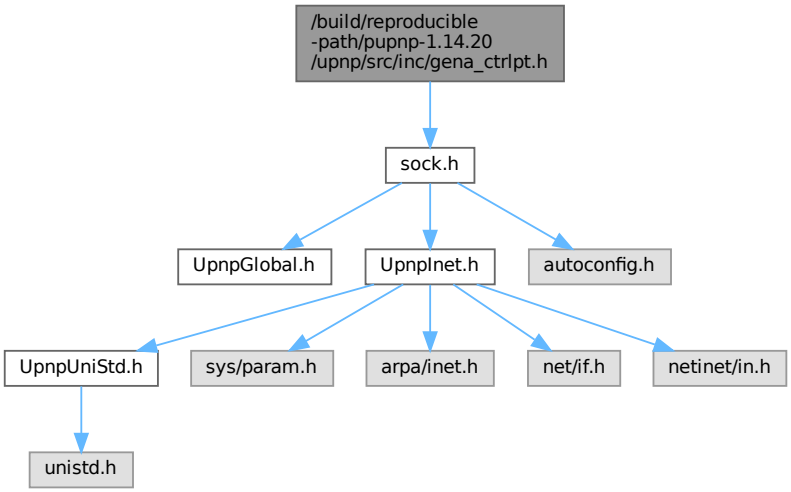
```
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENA_H
00033 #define GENA_H
00034
00039 #include "config.h"
00040
00041 #include <string.h>
00042 #include <time.h>
00043
00044 #include "ThreadPool.h"
00045 #include "UpnpString.h"
00046 #include "client_table.h"
00047 #include "httpparser.h"
00048 #include "miniserver.h"
00049 #include "service_table.h"
00050 #include "sock.h"
00051 #include "upnp.h"
00052 #include "uri.h"
00053
00054 #ifdef __cplusplus
00055         #define EXTERN_C extern "C"
00056 #else /* __cplusplus */
00057         #ifndef EXTERN_C
00058                 #define EXTERN_C
00059         #endif /* EXTERN_C */
00060 #endif         /* __cplusplus */
00061
00066 #define XML_VERSION "<?xml version='1.0' encoding='ISO-8859-1' ?>\n"
00067 #define XML_PROPERTYSET_HEADER \
00068         "<e:propertyset xmlns:e=\"urn:schemas-upnp-org:event-1-0\">\n"
00069
00070 #define UNABLE_MEMORY "HTTP/1.1 500 Internal Server Error\r\n\r\n"
00071 #define UNABLE_SERVICE_UNKNOWN "HTTP/1.1 404 Not Found\r\n\r\n"
00072 #define UNABLE_SERVICE_NOT_ACCEPT "HTTP/1.1 503 Service Not Available\r\n\r\n"
00073
00074 #define NOT_IMPLEMENTED "HTTP/1.1 501 Not Implemented\r\n\r\n"
00075 #define BAD_REQUEST "HTTP/1.1 400 Bad Request\r\n\r\n"
00076 #define INVALID_NT BAD_CALLBACK
00077 #define BAD_CALLBACK "HTTP/1.1 412 Precondition Failed\r\n\r\n"
00078 #define HTTP_OK_CRLF "HTTP/1.1 200 OK\r\n\r\n"
00079 #define HTTP_OK_STR "HTTP/1.1 200 OK\r\n"
00080 #define INVALID_SID BAD_CALLBACK
00081 #define MISSING_SID BAD_CALLBACK
00082 #define MAX_CONTENT_LENGTH 20
00083 #define MAX_SECONDS 10
00084 #define MAX_EVENTS 20
00085 #define MAX_PORT_SIZE 10
00086
00087 #define GENA_E_BAD_RESPONSE UPNP_E_BAD_RESPONSE
00088 #define GENA_E_BAD_SERVICE UPNP_E_INVALID_SERVICE
00089 #define GENA_E_SUBSCRIPTION_UNACCEPTED UPNP_E_SUBSCRIBE_UNACCEPTED
00090 #define GENA_E_BAD_SID UPNP_E_INVALID_SID
00091 #define GENA_E_UNSUBSCRIBE_UNACCEPTED UPNP_E_UNSUBSCRIBE_UNACCEPTED
00092 #define GENA_E_NOTIFY_UNACCEPTED UPNP_E_NOTIFY_UNACCEPTED
00093 #define GENA_E_NOTIFY_UNACCEPTED_REMOVE_SUB -9
00094 #define GENA_E_BAD_HANDLE UPNP_E_INVALID_HANDLE
00095
00096 #define XML_ERROR -5
00097 #define XML_SUCCESS UPNP_E_SUCCESS
00098 #define GENA_SUCCESS UPNP_E_SUCCESS
00099
00100 #define CALLBACK_SUCCESS 0
00101 #define DEFAULT_TIMEOUT 1801
00102
00103 extern ithread_mutex_t GlobalClientSubscribeMutex;
00104
00108 #define SubscribeLock() \
00109         UpnpPrintf( \
00110                 UPNP_INFO, GENA, __FILE__, __LINE__, "Trying Subscribe Lock"); \
00111         ithread_mutex_lock(&GlobalClientSubscribeMutex); \
00112         UpnpPrintf(UPNP_INFO, GENA, __FILE__, __LINE__, "Subscribe Lock");
00113
00117 #define SubscribeUnlock() \
00118         UpnpPrintf(UPNP_INFO, \
00119                 GENA, \
00120                 __FILE__, \
```

```
00121                 __LINE__, \
00122                 "Trying Subscribe UnLock"); \
00123         ithread_mutex_unlock(&GlobalClientSubscribeMutex); \
00124         UpnpPrintf(UPNP_INFO, GENA, __FILE__, __LINE__, "Subscribe UnLock");
00125
00129 typedef struct NOTIFY_THREAD_STRUCT
00130 {
00131         char *headers;
00132         DOMString propertySet;
00133         char *servId;
00134         char *UDN;
00135         Upnp_SID sid;
00136         time_t ctime;
00137         int *reference_count;
00138         UpnpDevice_Handle device_handle;
00139 } notify_thread_struct;
00140
00147 EXTERN_C void genaCallback(
00149         http_parser_t *parser,
00151         http_message_t *request,
00153         SOCKINFO *info);
00154
00165 #ifdef INCLUDE_CLIENT_APIS
00166 EXTERN_C int genaSubscribe(
00168         UpnpClient_Handle client_handle,
00170         const UpnpString *PublisherURL,
00175         int *TimeOut,
00177         UpnpString *out_sid);
00178 #endif /* INCLUDE_CLIENT_APIS */
00179
00190 #ifdef INCLUDE_CLIENT_APIS
00191 EXTERN_C int genaUnSubscribe(
00193         UpnpClient_Handle client_handle,
00195         const UpnpString *in_sid);
00196 #endif /* INCLUDE_CLIENT_APIS */
00197
00207 #ifdef INCLUDE_CLIENT_APIS
00208 EXTERN_C int genaUnregisterClient(
00210         UpnpClient_Handle client_handle);
00211 #endif /* INCLUDE_CLIENT_APIS */
00212
00213 /*
00214  * DEVICE
00215  */
00216
00222 #ifdef INCLUDE_DEVICE_APIS
00223 EXTERN_C int genaUnregisterDevice(
00225         UpnpDevice_Handle device_handle);
00226 #endif /* INCLUDE_CLIENT_APIS */
00227
00238 #ifdef INCLUDE_CLIENT_APIS
00239 EXTERN_C int genaRenewSubscription(
00241         UpnpClient_Handle client_handle,
00243         const UpnpString *in_sid,
00246         int *TimeOut);
00247 #endif /* INCLUDE_CLIENT_APIS */
00248
00257 #ifdef INCLUDE_DEVICE_APIS
00258 EXTERN_C int genaNotifyAll(
00260         UpnpDevice_Handle device_handle,
00262         char *UDN,
00264         char *servId,
00266         char **VarNames,
00268         char **VarValues,
00270         int var_count);
00271 #endif /* INCLUDE_DEVICE_APIS */
00272
00281 #ifdef INCLUDE_DEVICE_APIS
00282 EXTERN_C int genaNotifyAllExt(
00284         UpnpDevice_Handle device_handle,
00286         char *UDN,
00288         char *servId,
00290         IXML_Document *PropSet);
00291 #endif /* INCLUDE_DEVICE_APIS */
00292
00301 #ifdef INCLUDE_DEVICE_APIS
00302 EXTERN_C int genaInitNotify(
00304         UpnpDevice_Handle device_handle,
00306         char *UDN,
00308         char *servId,
00310         char **VarNames,
00312         char **VarValues,
00314         int var_count,
00316         const Upnp_SID sid);
00317 #endif /* INCLUDE_DEVICE_APIS */
00318
00329 #ifdef INCLUDE_DEVICE_APIS
```

```
00330 EXTERN_C int genaInitNotifyExt(
00332         UpnpDevice_Handle device_handle,
00334         char *UDN,
00336         char *servId,
00338         IXML_Document *PropSet,
00340         const Upnp_SID sid);
00341 #endif /* INCLUDE_DEVICE_APIS */
00342
00349 void error_respond(
00351         SOCKINFO *info,
00353         int error_code,
00355         http_message_t *hmsg);
00356
00357 #endif /* GENA_H */
```

9.102 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_ctrlpt.h File Reference

`#include "sock.h"`
Include dependency graph for gena_ctrlpt.h:



Functions

- void gena_process_notification_event (SOCKINFO *info, http_message_t *event)
  This function processes NOTIFY events that are sent by devices.

9.102.1 Function Documentation

9.102.1.1 gena_process_notification_event()

```
void gena_process_notification_event (
            SOCKINFO * info,
            http_message_t * event )
```

This function processes NOTIFY events that are sent by devices.

Parameters: IN SOCKINFO *info: Socket structure containing the device socket information IN http_message_t *event: The http message contains the GENA notification

Note

called by genacallback()

Parameters

in	*info*	Socket info of the device.
in	*event*	The http message contains the GENA notification.

References memptr::buf, Handle_Info::Callback, Handle_Info::ClientSubList, Handle_Info::Cookie, http_message_t::entity, GenlibClientSubscription_get_SID(), GetClientHandleInfo(), GetHandleInfo(), has_xml_content_type(), ixmlDocument_free(), ixmlParseBufferEx(), memptr::length, membuffer::length, http_message_t::msg, PARSE_OK, SubscribeLock, SubscribeUnlock, UPNP_EVENT_RECEIVED, UpnpEvent_delete(), UpnpEvent_new(), UpnpEvent_set_ChangedVariables(), UpnpEvent_set_EventKey(), and UpnpEvent_set_SID().

Referenced by genaCallback().

9.103 gena_ctrlpt.h

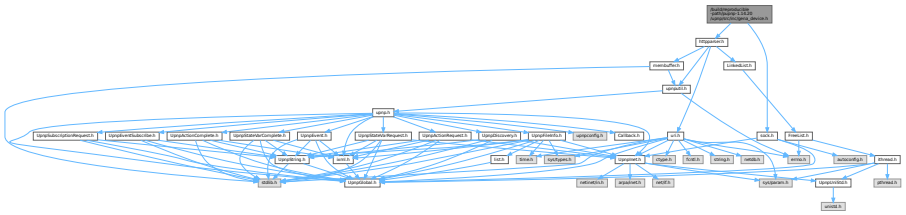
Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENA_CTRLPT_H
00033 #define GENA_CTRLPT_H
00034
00039 #include "sock.h"
00040
00052 void gena_process_notification_event(
00054         SOCKINFO *info,
00056         http_message_t *event);
00057
00058 #endif /* GENA_CTRLPT_H */
```

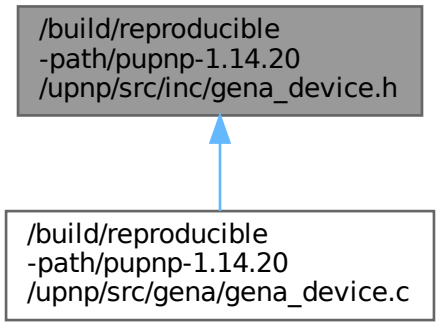
9.104 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_↩ device.h File Reference

```
#include "httpparser.h"
#include "sock.h"
```

Include dependency graph for gena_device.h:



This graph shows which files directly or indirectly include this file:



Functions

- void gena_process_subscription_request (SOCKINFO *info, http_message_t *request)

  Handles a subscription request from a ctrl point. The socket is not closed on return.
- void gena_process_subscription_renewal_request (SOCKINFO *info, http_message_t *request)

  Handles a subscription renewal request from a ctrl point. The connection is not destroyed on return.
- void gena_process_unsubscribe_request (SOCKINFO *info, http_message_t *request)

  Handles a subscription cancellation request from a ctrl point. The connection is not destroyed on return.

9.104.1 Function Documentation

9.104.1.1 gena_process_subscription_renewal_request()

```
void gena_process_subscription_renewal_request (
            SOCKINFO * info,
            http_message_t * request )
```

Handles a subscription renewal request from a ctrl point. The connection is not destroyed on return.

Parameters

in	*info*	Socket info of the device.
in	*request*	Subscription renewal request from the control point.

References memptr::buf, membuffer::buf, device_handle, SOCKINFO::foreign_sockaddr, GetDeviceHandleInfoForPath(), memptr::length, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, PARSE_OK, respond_ok(), UPNP_E_SUCCESS, UpnpPrintf(), and http_message_t::uri.

Referenced by genaCallback().

9.104.1.2 gena_process_subscription_request()

```
void gena_process_subscription_request (
            SOCKINFO * info,
            http_message_t * request )
```

Handles a subscription request from a ctrl point. The socket is not closed on return.

Parameters

in	*info*	Socket info of the device.
in	*request*	Subscription request from the control point.

References memptr::buf, Handle_Info::Callback, Handle_Info::Cookie, create_url_list(), device_handle, SOCKINFO::foreign_sockaddr, gena_validate_delivery_urls(), GetDeviceHandleInfoForPath(), memptr::length, Handle_Info::MaxSubscriptions, Handle_Info::MaxSubscriptionTimeOut, PARSE_OK, respond_ok(), UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, UPNP_EVENT_SUBSCRIPTION_REQUEST, UpnpPrintf(), UpnpSubscriptionRequest_delete(), UpnpSubscriptionRequest_new(), UpnpSubscriptionRequest_strcpy_ServiceId(), UpnpSubscriptionRequest_strcpy_SID(), UpnpSubscriptionRequest_strcpy_UDN(), http_message_t::uri, and URL_LIST::URLs.

Referenced by genaCallback().

9.104.1.3 gena_process_unsubscribe_request()

```
void gena_process_unsubscribe_request (
            SOCKINFO * info,
            http_message_t * request )
```

Handles a subscription cancellation request from a ctrl point. The connection is not destroyed on return.

Parameters

in	*info*	Socket info of the device.
in	*request*	UNSUBSCRIBE request from the control point.

References memptr::buf, membuffer::buf, device_handle, SOCKINFO::foreign_sockaddr, GetDeviceHandleInfoForPath(), memptr::length, and http_message_t::uri.

Referenced by genaCallback().

9.105 gena_device.h

Go to the documentation of this file.

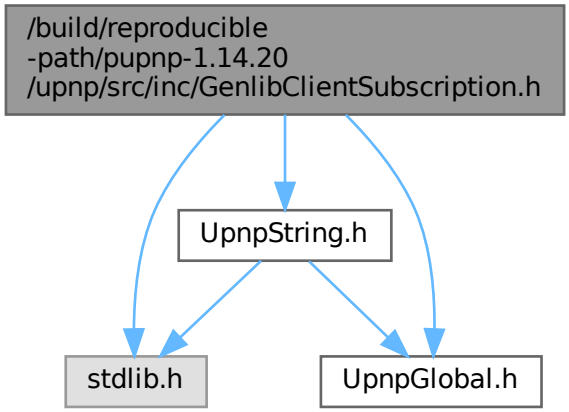
```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * * Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * * Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * * Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENA_DEVICE_H
00033 #define GENA_DEVICE_H
00034
00039 #include "httpparser.h"
00040 #include "sock.h"
00041
00046 void gena_process_subscription_request(
00048         SOCKINFO *info,
00050         http_message_t *request);
00051
00056 void gena_process_subscription_renewal_request(
00058         SOCKINFO *info,
00060         http_message_t *request);
00061
00066 void gena_process_unsubscribe_request(
00068         SOCKINFO *info,
00070         http_message_t *request);
00071
00072 #endif /* GENA_DEVICE_H */
```

9.106 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/GenlibClient↩ Subscription.h File Reference

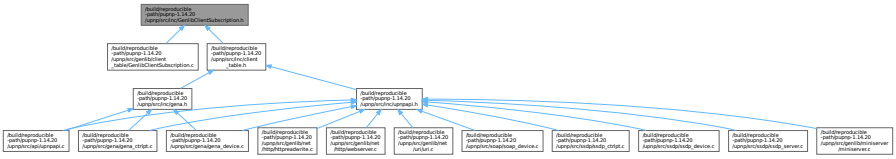
Header file for GenlibClientSubscription methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "UpnpString.h"
```

Include dependency graph for GenlibClientSubscription.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_GenlibClientSubscription GenlibClientSubscription

Functions

- UPNP_EXPORT_SPEC GenlibClientSubscription * GenlibClientSubscription_new ()
- UPNP_EXPORT_SPEC void GenlibClientSubscription_delete (GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC GenlibClientSubscription * GenlibClientSubscription_dup (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_assign (GenlibClientSubscription *p, const GenlibClientSubscription *q)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_get_RenewEventId (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_set_RenewEventId (GenlibClientSubscription *p, int n)
- UPNP_EXPORT_SPEC const UpnpString * GenlibClientSubscription_get_SID (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_set_SID (GenlibClientSubscription *p, const UpnpString *s)
- UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_SID_Length (const GenlibClientSubscription *p)

- UPNP_EXPORT_SPEC const char * GenlibClientSubscription_get_SID_cstr (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_SID (GenlibClientSubscription *p, const char *s)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_SID (GenlibClientSubscription *p, const char *s, size_t n)
- UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_SID (GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC const UpnpString * GenlibClientSubscription_get_ActualSID (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_set_ActualSID (GenlibClientSubscription *p, const UpnpString *s)
- UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_ActualSID_Length (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC const char * GenlibClientSubscription_get_ActualSID_cstr (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_ActualSID (GenlibClientSubscription *p, const char *s)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_ActualSID (GenlibClientSubscription *p, const char *s, size_t n)
- UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_ActualSID (GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC const UpnpString * GenlibClientSubscription_get_EventURL (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_set_EventURL (GenlibClientSubscription *p, const UpnpString *s)
- UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_EventURL_Length (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC const char * GenlibClientSubscription_get_EventURL_cstr (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_EventURL (GenlibClientSubscription *p, const char *s)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_EventURL (GenlibClientSubscription *p, const char *s, size_t n)
- UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_EventURL (GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC GenlibClientSubscription * GenlibClientSubscription_get_Next (const GenlibClientSubscription *p)
- UPNP_EXPORT_SPEC int GenlibClientSubscription_set_Next (GenlibClientSubscription *p, GenlibClientSubscription *n)

9.106.1 Detailed Description

Header file for GenlibClientSubscription methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.106.2 Typedef Documentation

9.106.2.1 GenlibClientSubscription

```
typedef struct s_GenlibClientSubscription GenlibClientSubscription
```

GenlibClientSubscription

9.106.3 Function Documentation

9.106.3.1 GenlibClientSubscription_assign()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_assign (
            GenlibClientSubscription * p,
            const GenlibClientSubscription * q )
```

Assignment operator

References GenlibClientSubscription_get_ActualSID(), GenlibClientSubscription_get_EventURL(), GenlibClientSubscription_get_Nex GenlibClientSubscription_get_RenewEventId(), GenlibClientSubscription_get_SID(), GenlibClientSubscription_set_ActualSID(), GenlibClientSubscription_set_EventURL(), GenlibClientSubscription_set_Next(), GenlibClientSubscription_set_RenewEventId(), and GenlibClientSubscription_set_SID().

Referenced by genaRenewSubscription(), genaUnregisterClient(), genaUnSubscribe(), and GenlibClientSubscription_dup().

9.106.3.2 GenlibClientSubscription_clear_ActualSID()

```
UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_ActualSID (
            GenlibClientSubscription * p )
```

GenlibClientSubscription_clear_ActualSID

References UpnpString_clear().

9.106.3.3 GenlibClientSubscription_clear_EventURL()

```
UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_EventURL (
            GenlibClientSubscription * p )
```

GenlibClientSubscription_clear_EventURL

References UpnpString_clear().

9.106.3.4 GenlibClientSubscription_clear_SID()

```
UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_SID (
            GenlibClientSubscription * p )
```

GenlibClientSubscription_clear_SID

References UpnpString_clear().

9.106.3.5 GenlibClientSubscription_delete()

```
UPNP_EXPORT_SPEC void GenlibClientSubscription_delete (
            GenlibClientSubscription * p )
```

Destructor

References UpnpString_delete().

Referenced by freeClientSubList(), genaRenewSubscription(), genaSubscribe(), genaUnregisterClient(), and genaUnSubscribe().

9.106.3.6 GenlibClientSubscription_dup()

```
UPNP_EXPORT_SPEC GenlibClientSubscription * GenlibClientSubscription_dup (
            const GenlibClientSubscription * p )
```

Copy Constructor

References GenlibClientSubscription_assign(), and GenlibClientSubscription_new().

9.106.3.7 GenlibClientSubscription_get_ActualSID()

```
UPNP_EXPORT_SPEC const UpnpString * GenlibClientSubscription_get_ActualSID (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_ActualSID

Referenced by genaRenewSubscription(), genaUnregisterClient(), genaUnSubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_get_ActualSID_cstr(), and GenlibClientSubscription_get_ActualSID_Length().

9.106.3.8 GenlibClientSubscription_get_ActualSID_cstr()

```
UPNP_EXPORT_SPEC const char * GenlibClientSubscription_get_ActualSID_cstr (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_ActualSID_cstr

References GenlibClientSubscription_get_ActualSID(), and UpnpString_get_String().

Referenced by GetClientSubActualSID().

9.106.3.9 GenlibClientSubscription_get_ActualSID_Length()

```
UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_ActualSID_Length (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_ActualSID_Length

References GenlibClientSubscription_get_ActualSID(), and UpnpString_get_Length().

9.106.3.10 GenlibClientSubscription_get_EventURL()

```
UPNP_EXPORT_SPEC const UpnpString * GenlibClientSubscription_get_EventURL (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_EventURL

Referenced by genaRenewSubscription(), genaUnregisterClient(), genaUnSubscribe(), GenlibClientSubscription_assign(), GenlibClientSubscription_get_EventURL_cstr(), GenlibClientSubscription_get_EventURL_Length(), and ScheduleGenaAutoRenew().

9.106.3.11 GenlibClientSubscription_get_EventURL_cstr()

```
UPNP_EXPORT_SPEC const char * GenlibClientSubscription_get_EventURL_cstr (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_EventURL_cstr

References GenlibClientSubscription_get_EventURL(), and UpnpString_get_String().

9.106.3.12 GenlibClientSubscription_get_EventURL_Length()

```
UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_EventURL_Length (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_EventURL_Length

References GenlibClientSubscription_get_EventURL(), and UpnpString_get_Length().

9.106.3.13 GenlibClientSubscription_get_Next()

```
UPNP_EXPORT_SPEC GenlibClientSubscription * GenlibClientSubscription_get_Next (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_Next

Referenced by freeClientSubList(), GenlibClientSubscription_assign(), GetClientSubActualSID(), GetClientSubClientSID(), and RemoveClientSubClientSID().

9.106.3.14 GenlibClientSubscription_get_RenewEventId()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_get_RenewEventId (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_RenewEventId

Referenced by free_client_subscription(), genaRenewSubscription(), and GenlibClientSubscription_assign().

9.106.3.15 GenlibClientSubscription_get_SID()

```
UPNP_EXPORT_SPEC const UpnpString * GenlibClientSubscription_get_SID (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_SID

Referenced by gena_process_notification_event(), genaRenewSubscription(), genaUnregisterClient(), GenlibClientSubscription_assig GenlibClientSubscription_get_SID_cstr(), GenlibClientSubscription_get_SID_Length(), and ScheduleGenaAutoRenew().

9.106.3.16 GenlibClientSubscription_get_SID_cstr()

```
UPNP_EXPORT_SPEC const char * GenlibClientSubscription_get_SID_cstr (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_SID_cstr

References GenlibClientSubscription_get_SID(), and UpnpString_get_String().

Referenced by GetClientSubClientSID(), and RemoveClientSubClientSID().

9.106.3.17 GenlibClientSubscription_get_SID_Length()

```
UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_SID_Length (
            const GenlibClientSubscription * p )
```

GenlibClientSubscription_get_SID_Length

References GenlibClientSubscription_get_SID(), and UpnpString_get_Length().

9.106.3.18 GenlibClientSubscription_new()

```
UPNP_EXPORT_SPEC GenlibClientSubscription * GenlibClientSubscription_new ( )
```

Constructor

References UpnpString_new().

Referenced by genaRenewSubscription(), genaSubscribe(), genaUnregisterClient(), genaUnSubscribe(), and GenlibClientSubscription_dup().

9.106.3.19 GenlibClientSubscription_set_ActualSID()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_set_ActualSID (
            GenlibClientSubscription * p,
            const UpnpString * s )
```

GenlibClientSubscription_set_ActualSID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by genaRenewSubscription(), genaSubscribe(), and GenlibClientSubscription_assign().

9.106.3.20 GenlibClientSubscription_set_EventURL()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_set_EventURL (
            GenlibClientSubscription * p,
            const UpnpString * s )
```

GenlibClientSubscription_set_EventURL

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by genaSubscribe(), and GenlibClientSubscription_assign().

9.106.3.21 GenlibClientSubscription_set_Next()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_set_Next (
            GenlibClientSubscription * p,
            GenlibClientSubscription * n )
```

GenlibClientSubscription_set_Next

Referenced by genaSubscribe(), GenlibClientSubscription_assign(), and RemoveClientSubClientSID().

9.106.3.22 GenlibClientSubscription_set_RenewEventId()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_set_RenewEventId (
            GenlibClientSubscription * p,
            int n )
```

GenlibClientSubscription_set_RenewEventId

Referenced by free_client_subscription(), genaRenewSubscription(), genaSubscribe(), GenlibClientSubscription_assign(), and ScheduleGenaAutoRenew().

9.106.3.23 GenlibClientSubscription_set_SID()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_set_SID (
            GenlibClientSubscription * p,
            const UpnpString * s )
```

GenlibClientSubscription_set_SID

References UpnpString_get_String(), and UpnpString_set_String().

Referenced by genaSubscribe(), and GenlibClientSubscription_assign().

9.106.3.24 GenlibClientSubscription_strcpy_ActualSID()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_ActualSID (
            GenlibClientSubscription * p,
            const char * s )
```

GenlibClientSubscription_strcpy_ActualSID

References UpnpString_set_String().

Referenced by free_client_subscription().

9.106.3.25 GenlibClientSubscription_strcpy_EventURL()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_EventURL (
            GenlibClientSubscription * p,
            const char * s )
```

GenlibClientSubscription_strcpy_EventURL

References UpnpString_set_String().

Referenced by free_client_subscription().

9.106.3.26 GenlibClientSubscription_strcpy_SID()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_SID (
            GenlibClientSubscription * p,
            const char * s )
```

GenlibClientSubscription_strcpy_SID

References UpnpString_set_String().

9.106.3.27 GenlibClientSubscription_strncpy_ActualSID()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_ActualSID (
            GenlibClientSubscription * p,
            const char * s,
            size_t n )
```

GenlibClientSubscription_strncpy_ActualSID

References UpnpString_set_StringN().

9.106.3.28 GenlibClientSubscription_strncpy_EventURL()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_EventURL (
            GenlibClientSubscription * p,
            const char * s,
            size_t n )
```

GenlibClientSubscription_strncpy_EventURL

References UpnpString_set_StringN().

9.106.3.29 GenlibClientSubscription_strncpy_SID()

```
UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_SID (
            GenlibClientSubscription * p,
            const char * s,
            size_t n )
```

GenlibClientSubscription_strncpy_SID

References UpnpString_set_StringN().

9.107 GenlibClientSubscription.h

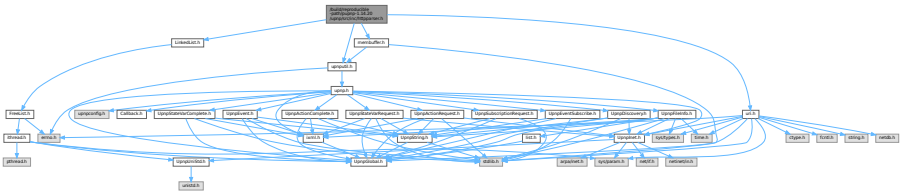
Go to the documentation of this file.

```
00001 #ifndef GENLIBCLIENTSUBSCRIPTION_H
00002 #define GENLIBCLIENTSUBSCRIPTION_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "UpnpString.h"
00019
00020 #ifdef __cplusplus
00021 extern "C" {
00022 #endif /* __cplusplus */
00023
00027 typedef struct s_GenlibClientSubscription GenlibClientSubscription;
00028
00030 UPNP_EXPORT_SPEC GenlibClientSubscription *GenlibClientSubscription_new();
00032 UPNP_EXPORT_SPEC void GenlibClientSubscription_delete(
00033         GenlibClientSubscription *p);
00035 UPNP_EXPORT_SPEC GenlibClientSubscription *GenlibClientSubscription_dup(
00036         const GenlibClientSubscription *p);
00038 UPNP_EXPORT_SPEC int GenlibClientSubscription_assign(
00039         GenlibClientSubscription *p, const GenlibClientSubscription *q);
00040
00042 UPNP_EXPORT_SPEC int GenlibClientSubscription_get_RenewEventId(
00043         const GenlibClientSubscription *p);
00045 UPNP_EXPORT_SPEC int GenlibClientSubscription_set_RenewEventId(
00046         GenlibClientSubscription *p, int n);
00047
00049 UPNP_EXPORT_SPEC const UpnpString *GenlibClientSubscription_get_SID(
00050         const GenlibClientSubscription *p);
00052 UPNP_EXPORT_SPEC int GenlibClientSubscription_set_SID(
00053         GenlibClientSubscription *p, const UpnpString *s);
00055 UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_SID_Length(
00056         const GenlibClientSubscription *p);
00058 UPNP_EXPORT_SPEC const char *GenlibClientSubscription_get_SID_cstr(
00059         const GenlibClientSubscription *p);
00061 UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_SID(
00062         GenlibClientSubscription *p, const char *s);
00064 UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_SID(
00065         GenlibClientSubscription *p, const char *s, size_t n);
00067 UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_SID(
00068         GenlibClientSubscription *p);
00069
00071 UPNP_EXPORT_SPEC const UpnpString *GenlibClientSubscription_get_ActualSID(
00072         const GenlibClientSubscription *p);
00074 UPNP_EXPORT_SPEC int GenlibClientSubscription_set_ActualSID(
00075         GenlibClientSubscription *p, const UpnpString *s);
00077 UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_ActualSID_Length(
00078         const GenlibClientSubscription *p);
00080 UPNP_EXPORT_SPEC const char *GenlibClientSubscription_get_ActualSID_cstr(
00081         const GenlibClientSubscription *p);
00083 UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_ActualSID(
00084         GenlibClientSubscription *p, const char *s);
00086 UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_ActualSID(
00087         GenlibClientSubscription *p, const char *s, size_t n);
00089 UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_ActualSID(
00090         GenlibClientSubscription *p);
00091
00093 UPNP_EXPORT_SPEC const UpnpString *GenlibClientSubscription_get_EventURL(
00094         const GenlibClientSubscription *p);
00096 UPNP_EXPORT_SPEC int GenlibClientSubscription_set_EventURL(
00097         GenlibClientSubscription *p, const UpnpString *s);
00099 UPNP_EXPORT_SPEC size_t GenlibClientSubscription_get_EventURL_Length(
00100         const GenlibClientSubscription *p);
00102 UPNP_EXPORT_SPEC const char *GenlibClientSubscription_get_EventURL_cstr(
00103         const GenlibClientSubscription *p);
00105 UPNP_EXPORT_SPEC int GenlibClientSubscription_strcpy_EventURL(
00106         GenlibClientSubscription *p, const char *s);
00108 UPNP_EXPORT_SPEC int GenlibClientSubscription_strncpy_EventURL(
00109         GenlibClientSubscription *p, const char *s, size_t n);
00111 UPNP_EXPORT_SPEC void GenlibClientSubscription_clear_EventURL(
00112         GenlibClientSubscription *p);
00113
00115 UPNP_EXPORT_SPEC GenlibClientSubscription *GenlibClientSubscription_get_Next(
00116         const GenlibClientSubscription *p);
00118 UPNP_EXPORT_SPEC int GenlibClientSubscription_set_Next(
00119         GenlibClientSubscription *p, GenlibClientSubscription *n);
00120
00121 #ifdef __cplusplus
00122 }
00123 #endif /* __cplusplus */
00124
00125 #endif /* GENLIBCLIENTSUBSCRIPTION_H */
```

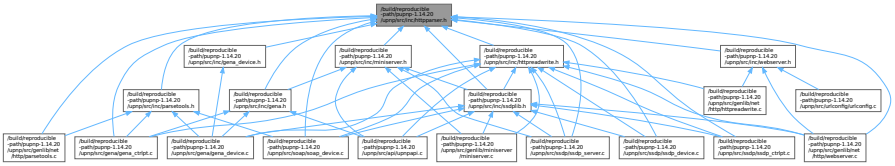
9.108 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h File Reference

```
#include "LinkedList.h"
#include "membuffer.h"
#include "upnputil.h"
#include "uri.h"
```

Include dependency graph for httpparser.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct scanner_t
- struct http_header_t
- struct http_message_t
- struct http_parser_t

Macros

- #define **ENTREAD_DETERMINE_READ_METHOD** 1
- #define **ENTREAD_USING_CLEN** 2
- #define **ENTREAD_USING_CHUNKED** 3
- #define **ENTREAD_UNTIL_CLOSE** 4
- #define **ENTREAD_CHUNKY_BODY** 5
- #define **ENTREAD_CHUNKY_HEADERS** 6
- #define **HDR_UNKNOWN** -1
- #define **HDR_CACHE_CONTROL** 1
- #define **HDR_CALLBACK** 2
- #define **HDR_CONTENT_LENGTH** 3
- #define **HDR_CONTENT_TYPE** 4
- #define **HDR_DATE** 5
- #define **HDR_EXT** 6
- #define **HDR_HOST** 7
- #define **HDR_LOCATION** 11

- #define **HDR_MAN** 12
- #define **HDR_MX** 13
- #define **HDR_NT** 14
- #define **HDR_NTS** 15
- #define **HDR_SERVER** 16
- #define **HDR_SEQ** 17
- #define **HDR_SID** 18
- #define **HDR_SOAPACTION** 19
- #define **HDR_ST** 20
- #define **HDR_TIMEOUT** 21
- #define **HDR_TRANSFER_ENCODING** 22
- #define **HDR_USN** 23
- #define **HDR_USER_AGENT** 24
- #define **HDR_ACCEPT** 25
- #define **HDR_ACCEPT_ENCODING** 26
- #define **HDR_ACCEPT_CHARSET** 27
- #define **HDR_ACCEPT_LANGUAGE** 28
- #define **HDR_ACCEPT_RANGE** 29
- #define **HDR_CONTENT_ENCODING** 30
- #define **HDR_CONTENT_LANGUAGE** 31
- #define **HDR_CONTENT_LOCATION** 32
- #define **HDR_CONTENT_RANGE** 33
- #define **HDR_IF_RANGE** 34
- #define **HDR_RANGE** 35
- #define **HDR_TE** 36

Enumerations

- enum **token_type_t** {
  TT_IDENTIFIER , **TT_WHITESPACE** , **TT_CRLF** , **TT_CTRL** ,
  TT_SEPARATOR , **TT_QUOTEDSTRING** }
- enum **parser_pos_t** {
  POS_REQUEST_LINE , **POS_RESPONSE_LINE** , **POS_HEADERS** , **POS_ENTITY** ,
  POS_COMPLETE }
- enum **http_method_t** {
  HTTPMETHOD_PUT = UPNP_HTTPMETHOD_PUT , **HTTPMETHOD_DELETE** = UPNP_HTTPMETHOD↩
  _DELETE , **HTTPMETHOD_GET** = UPNP_HTTPMETHOD_GET , **HTTPMETHOD_HEAD** = UPNP_↩
  HTTPMETHOD_HEAD ,
  HTTPMETHOD_POST = UPNP_HTTPMETHOD_POST , **HTTPMETHOD_MPOST** , **HTTPMETHOD_↩
  SUBSCRIBE** , **HTTPMETHOD_UNSUBSCRIBE** ,
  HTTPMETHOD_NOTIFY , **HTTPMETHOD_MSEARCH** , **HTTPMETHOD_UNKNOWN** , **SOAPMETHOD_↩
  POST** ,
  HTTPMETHOD_SIMPLEGET }
- enum parse_status_t {
  PARSE_SUCCESS = 0 , PARSE_INCOMPLETE , PARSE_INCOMPLETE_ENTITY , PARSE_FAILURE ,
  PARSE_OK , PARSE_NO_MATCH , PARSE_CONTINUE_1 }

Functions

- void **httpmsg_init** (http_message_t *msg)
- void **httpmsg_destroy** (http_message_t *msg)
- http_header_t * **httpmsg_find_hdr_str** (http_message_t *msg, const char *header_name)
- http_header_t * **httpmsg_find_hdr** (http_message_t *msg, int header_name_id, memptr *value)
- void **parser_request_init** (http_parser_t *parser)

- void **parser_response_init** (http_parser_t *parser, http_method_t request_method)
- parse_status_t **parser_parse** (http_parser_t *parser)
- parse_status_t **parser_parse_responseline** (http_parser_t *parser)
- parse_status_t **parser_parse_headers** (http_parser_t *parser)
- parse_status_t **parser_parse_entity** (http_parser_t *parser)
- parse_status_t **parser_get_entity_read_method** (http_parser_t *parser)
- parse_status_t **parser_append** (http_parser_t *parser, const char *buf, size_t buf_length)
- parse_status_t **matchstr** (char *str, size_t slen, const char *fmt,...)
- int **raw_to_int** (memptr *raw_value, int base)
- int **raw_find_str** (memptr *raw_value, const char *str)
- const char * **method_to_str** (http_method_t method)
- void print_http_headers (http_message_t *hmsg)

  Print the HTTP headers.

9.108.1 Enumeration Type Documentation

9.108.1.1 parse_status_t

```
enum parse_status_t
```

status of parsing

Enumerator

PARSE_SUCCESS	msg was parsed successfully.
PARSE_INCOMPLETE	need more data to continue.
PARSE_INCOMPLETE_ENTITY	for responses that don't have length specified.
PARSE_FAILURE	parse failed; check status code for details.
PARSE_OK	done partial.
PARSE_NO_MATCH	token not matched.
PARSE_CONTINUE_1	private.

9.108.2 Function Documentation

9.108.2.1 print_http_headers()

```
void print_http_headers (
            http_message_t * hmsg )
```

Print the HTTP headers.

Parameters

in	*hmsg*	HTTP Message object.

References memptr::buf, membuffer::buf, http_message_t::headers, http_message_t::is_request, memptr::length, membuffer::length, http_message_t::method, http_header_t::name, http_message_t::status_code, http_message_t::status_msg, UpnpPrintf(), http_message_t::uri, and http_header_t::value.

9.109 httpparser.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (c) 2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * - Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * - Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * - Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef GENLIB_NET_HTTP_HTTPPARSER_H
00034 #define GENLIB_NET_HTTP_HTTPPARSER_H
00035
00040 #include "LinkedList.h"
00041 #include "membuffer.h"
00042 #include "upnputil.h"
00043 #include "uri.h"
00044
00045 /* private types */
00046
00047 /* scanner */
00048
00049 /* Used to represent different types of tokens in input. */
00050 typedef enum
00051 {
00052         TT_IDENTIFIER,
00053         TT_WHITESPACE,
00054         TT_CRLF,
00055         TT_CTRL,
00056         TT_SEPARATOR,
00057         TT_QUOTEDSTRING
00058 } token_type_t;
00059
00060 typedef struct
00061 {
00063         membuffer *msg;
00065         size_t cursor;
00068         int entire_msg_loaded;
00069 } scanner_t;
00070
00071 typedef enum
00072 {
00073         POS_REQUEST_LINE,
00074         POS_RESPONSE_LINE,
00075         POS_HEADERS,
00076         POS_ENTITY,
00077         POS_COMPLETE
00078 } parser_pos_t;
00079
00080 #define ENTREAD_DETERMINE_READ_METHOD 1
00081 #define ENTREAD_USING_CLEN 2
00082 #define ENTREAD_USING_CHUNKED 3
00083 #define ENTREAD_UNTIL_CLOSE 4
00084 #define ENTREAD_CHUNKY_BODY 5
00085 #define ENTREAD_CHUNKY_HEADERS 6
00086
00087 /* end of private section. */
00088
00089 /* method in a HTTP request.
00090  * IMPORTANT: The enum values of the standard HTTP method should match
```

```
00091  * those of Upnp_HttpMethod enum defined in upnp.h */
00092 typedef enum
00093 {
00094         HTTPMETHOD_PUT = UPNP_HTTPMETHOD_PUT,
00095         HTTPMETHOD_DELETE = UPNP_HTTPMETHOD_DELETE,
00096         HTTPMETHOD_GET = UPNP_HTTPMETHOD_GET,
00097         HTTPMETHOD_HEAD = UPNP_HTTPMETHOD_HEAD,
00098         HTTPMETHOD_POST = UPNP_HTTPMETHOD_POST,
00099         HTTPMETHOD_MPOST,
00100         HTTPMETHOD_SUBSCRIBE,
00101         HTTPMETHOD_UNSUBSCRIBE,
00102         HTTPMETHOD_NOTIFY,
00103         HTTPMETHOD_MSEARCH,
00104         HTTPMETHOD_UNKNOWN,
00105         SOAPMETHOD_POST,
00106         HTTPMETHOD_SIMPLEGET
00107 } http_method_t;
00108
00109 /* different types of HTTP headers */
00110 #define HDR_UNKNOWN -1
00111 #define HDR_CACHE_CONTROL 1
00112 #define HDR_CALLBACK 2
00113 #define HDR_CONTENT_LENGTH 3
00114 #define HDR_CONTENT_TYPE 4
00115 #define HDR_DATE 5
00116 #define HDR_EXT 6
00117 #define HDR_HOST 7
00118 /*define HDR_IF_MODIFIED_SINCE          8 */
00119 /*define HDR_IF_UNMODIFIED_SINCE        9 */
00120 /*define HDR_LAST_MODIFIED              10 */
00121 #define HDR_LOCATION 11
00122 #define HDR_MAN 12
00123 #define HDR_MX 13
00124 #define HDR_NT 14
00125 #define HDR_NTS 15
00126 #define HDR_SERVER 16
00127 #define HDR_SEQ 17
00128 #define HDR_SID 18
00129 #define HDR_SOAPACTION 19
00130 #define HDR_ST 20
00131 #define HDR_TIMEOUT 21
00132 #define HDR_TRANSFER_ENCODING 22
00133 #define HDR_USN 23
00134 #define HDR_USER_AGENT 24
00135
00136 /* Adding new header difinition */
00137 #define HDR_ACCEPT 25
00138 #define HDR_ACCEPT_ENCODING 26
00139 #define HDR_ACCEPT_CHARSET 27
00140 #define HDR_ACCEPT_LANGUAGE 28
00141 #define HDR_ACCEPT_RANGE 29
00142 #define HDR_CONTENT_ENCODING 30
00143 #define HDR_CONTENT_LANGUAGE 31
00144 #define HDR_CONTENT_LOCATION 32
00145 #define HDR_CONTENT_RANGE 33
00146 #define HDR_IF_RANGE 34
00147 #define HDR_RANGE 35
00148 #define HDR_TE 36
00149
00151 typedef enum
00152 {
00154         PARSE_SUCCESS = 0,
00156         PARSE_INCOMPLETE,
00158         PARSE_INCOMPLETE_ENTITY,
00160         PARSE_FAILURE,
00162         PARSE_OK,
00164         PARSE_NO_MATCH,
00166         PARSE_CONTINUE_1
00167 } parse_status_t;
00168
00169 typedef struct
00170 {
00172         memptr name;
00174         int name_id;
00176         membuffer value;
00177         /* private. */
00178         membuffer name_buf;
00179 } http_header_t;
00180
00181 typedef struct
00182 {
00183         int initialized;
00185         http_method_t method;
00187         uri_type uri;
00189         http_method_t request_method;
00191         int status_code;
00193         membuffer status_msg;
```

```
00197         size_t amount_discarded;
00198         /* fields used in both request or response messages. */
00200         int is_request;
00201         /* http major version. */
00202         int major_version;
00203         /* http minor version. */
00204         int minor_version;
00206         LinkedList headers;
00208         memptr entity;
00209         /* private fields. */
00211         membuffer msg;
00213         char *urlbuf;
00214 } http_message_t;
00215
00216 typedef struct
00217 {
00218         http_message_t msg;
00221         int http_error_code;
00224         int valid_ssdp_notify_hack;
00225         /* private data -- don't touch. */
00226         parser_pos_t position;
00227         int ent_position;
00228         unsigned int content_length;
00229         size_t chunk_size;
00232         size_t entity_start_position;
00233         scanner_t scanner;
00234 } http_parser_t;
00235
00236 #ifdef __cplusplus
00237 extern "C" {
00238 #endif /* __cplusplus */
00239
00240 /************************************************************************
00241  *      Function :      httpmsg_init
00242  *
00243  *      Parameters :
00244  *              INOUT http_message_t* msg ;     HTTP Message Object
00245  *
00246  *      Description :   Initialize and allocate memory for http message
00247  *
00248  *      Return : void ;
00249  *
00250  *      Note :
00251  ************************************************************************/
00252 void httpmsg_init(http_message_t *msg);
00253
00254 /************************************************************************
00255  *      Function :      httpmsg_destroy
00256  *
00257  *      Parameters :
00258  *              INOUT http_message_t* msg ;     HTTP Message Object
00259  *
00260  *      Description :   Free memory allocated for the http message
00261  *
00262  *      Return : void ;
00263  *
00264  *      Note :
00265  ************************************************************************/
00266 void httpmsg_destroy(http_message_t *msg);
00267
00268 /************************************************************************
00269  *      Function :      httpmsg_find_hdr_str
00270  *
00271  *      Parameters :
00272  *              IN http_message_t* msg ;        HTTP Message Object
00273  *              IN const char* header_name ; Header name to be compared with
00274  *
00275  *      Description :   Compares the header name with the header names stored
00276  *              in      the linked list of messages
00277  *
00278  *      Return : http_header_t* - Pointer to a header on success;
00279  *                       NULL on failure
00280  *      Note :
00281  ************************************************************************/
00282 http_header_t *httpmsg_find_hdr_str(
00283         http_message_t *msg, const char *header_name);
00284
00285 /************************************************************************
00286  *      Function :      httpmsg_find_hdr
00287  *
00288  *      Parameters :
00289  *              IN http_message_t* msg ; HTTP Message Object
00290  *              IN int header_name_id ;  Header Name ID to be compared with
00291  *              OUT memptr* value ;              Buffer to get the ouput to.
00292  *
00293  *      Description :   Finds header from a list, with the given 'name_id'.
00294  *
```

```
00295  *      Return : http_header_t*  - Pointer to a header on success;
00296  *               NULL on failure
00297  *
00298  *      Note :
00299  ***************************************************************************/
00300 http_header_t *httpmsg_find_hdr(
00301         http_message_t *msg, int header_name_id, memptr *value);
00302
00303 /***************************************************************************
00304  * Function: parser_request_init
00305  *
00306  * Parameters:
00307  *      OUT http_parser_t* parser ; HTTP Parser object
00308  *
00309  * Description: Initializes parser object for a request
00310  *
00311  * Returns:
00312  *       void
00313  ***************************************************************************/
00314 void parser_request_init(http_parser_t *parser);
00315
00316 /***************************************************************************
00317  * Function: parser_response_init
00318  *
00319  * Parameters:
00320  *      OUT http_parser_t* parser       ;         HTTP Parser object
00321  *      IN http_method_t request_method ; Request method
00322  *
00323  * Description: Initializes parser object for a response
00324  *
00325  * Returns:
00326  *       void
00327  ***************************************************************************/
00328 void parser_response_init(http_parser_t *parser, http_method_t request_method);
00329
00330 /***************************************************************************
00331  * Function: parser_parse
00332  *
00333  * Parameters:
00334  *      INOUT http_parser_t* parser ; HTTP Parser object
00335  *
00336  * Description: The parser function. Depending on the position of the
00337  *      parser object the actual parsing function is invoked
00338  *
00339  * Returns:
00340  *       void
00341  ***************************************************************************/
00342 parse_status_t parser_parse(http_parser_t *parser);
00343
00344 /***************************************************************************
00345  * Function: parser_parse_responseline
00346  *
00347  * Parameters:
00348  *      INOUT http_parser_t* parser     ; HTTP Parser object
00349  *
00350  * Description: Get HTTP Method, URL location and version information.
00351  *
00352  * Returns:
00353  *      PARSE_OK
00354  *      PARSE_SUCCESS
00355  *      PARSE_FAILURE
00356  ***************************************************************************/
00357 parse_status_t parser_parse_responseline(http_parser_t *parser);
00358
00359 /***************************************************************************
00360  * Function: parser_parse_headers
00361  *
00362  * Parameters:
00363  *      INOUT http_parser_t* parser     ; HTTP Parser object
00364  *
00365  * Description: Get HTTP Method, URL location and version information.
00366  *
00367  * Returns:
00368  *      PARSE_OK
00369  *      PARSE_SUCCESS
00370  *      PARSE_FAILURE
00371  ***************************************************************************/
00372 parse_status_t parser_parse_headers(http_parser_t *parser);
00373
00374 /***************************************************************************
00375  * Function: parser_parse_entity
00376  *
00377  * Parameters:
00378  *      INOUT http_parser_t* parser     ; HTTP Parser object
00379  *
00380  * Description: Determines method to read entity
00381  *
```

```
00382  * Returns:
00383  *       PARSE_OK
00384  *       PARSE_FAILURE
00385  *       PARSE_COMPLETE -- no more reading to do
00386  ***************************************************************************/
00387 parse_status_t parser_parse_entity(http_parser_t *parser);
00388
00389 /***************************************************************************
00390  * Function: parser_get_entity_read_method
00391  *
00392  * Parameters:
00393  *      INOUT http_parser_t* parser     ; HTTP Parser object
00394  *
00395  * Description: Determines method to read entity
00396  *
00397  * Returns:
00398  *       PARSE_OK
00399  *       PARSE_FAILURE
00400  *       PARSE_COMPLETE -- no more reading to do
00401  ***************************************************************************/
00402 parse_status_t parser_get_entity_read_method(http_parser_t *parser);
00403
00404 /***************************************************************************
00405  * Function: parser_append
00406  *
00407  * Parameters:
00408  *      INOUT http_parser_t* parser ;   HTTP Parser Object
00409  *      IN const char* buf      ;       buffer to be appended to the parser
00410  *                                      buffer
00411  *      IN size_t buf_length ;          Size of the buffer
00412  *
00413  * Description: The parser function. Depending on the position of the
00414  *      parser object the actual parsing function is invoked
00415  *
00416  * Returns:
00417  *       void
00418  ***************************************************************************/
00419 parse_status_t parser_append(
00420         http_parser_t *parser, const char *buf, size_t buf_length);
00421
00422 /***************************************************************************
00423  * Function: matchstr
00424  *
00425  * Parameters:
00426  *      IN char *str ;           String to be matched
00427  *      IN size_t slen ;     Length of the string
00428  *      IN const char* fmt ; Pattern format
00429  *      ...
00430  *
00431  * Description: Matches a variable parameter list with a string
00432  *      and takes actions based on the data type specified.
00433  *
00434  * Returns:
00435  *   PARSE_OK
00436  *   PARSE_NO_MATCH -- failure to match pattern 'fmt'
00437  *   PARSE_FAILURE      -- 'str' is bad input
00438  ***************************************************************************/
00439 parse_status_t matchstr(char *str, size_t slen, const char *fmt, ...);
00440
00441 /***************************************************************************
00442  * Function: raw_to_int
00443  *
00444  * Parameters:
00445  *      IN memptr* raw_value ;  Buffer to be converted
00446  *      IN int base ;                   Base  to use for conversion
00447  *
00448  * Description: Converts raw character data to long-integer value
00449  *
00450  * Returns:
00451  *       int
00452  ***************************************************************************/
00453 int raw_to_int(memptr *raw_value, int base);
00454
00455 /***************************************************************************
00456  * Function: raw_find_str
00457  *
00458  * Parameters:
00459  *      IN memptr* raw_value ; Buffer containg the string
00460  *      IN const char* str ;    Substring to be found
00461  *
00462  * Description: Find a substring from raw character string buffer
00463  *
00464  * Side effects: raw_value is transformed to lowercase.
00465  *
00466  * Returns:
00467  *       int - index at which the substring is found.
00468  ***************************************************************************/
```

```
00469 int raw_find_str(memptr *raw_value, const char *str);
00470
00471 /************************************************************************
00472  * Function: method_to_str
00473  *
00474  * Parameters:
00475  * IN http_method_t method ; HTTP method
00476  *
00477  * Description: A wrapper function that maps a method id to a method
00478  *      nameConverts a http_method id stored in the HTTP Method
00479  *
00480  * Returns:
00481  *       const char* ptr - Ptr to the HTTP Method
00482  ************************************************************************/
00483 const char *method_to_str(http_method_t method);
00484
00489 #ifdef DEBUG
00490 void print_http_headers(
00492         http_message_t *hmsg);
00493 #else
00494         #define print_http_headers(hmsg) \
00495                 do { \
00496                 } while (0)
00497 #endif
00498
00499 #ifdef __cplusplus
00500 } /* extern "C" */
00501 #endif /* __cplusplus */
00502
00503 #endif /* GENLIB_NET_HTTP_HTTPPARSER_H */
```

9.110 httpreadwrite.h

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (c) 2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * - Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * - Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * - Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef GENLIB_NET_HTTP_HTTPREADWRITE_H
00034 #define GENLIB_NET_HTTP_HTTPREADWRITE_H
00035
00036 /*
00037  * \file
00038  */
00039
00040 #include "config.h"
00041 #include "httpparser.h"
00042 #include "sock.h"
00043 #include "upnputil.h"
00044
00046 #define HTTP_DEFAULT_TIMEOUT 30
00047
00048 #ifdef __cplusplus
00049 extern "C" {
00050 #endif
00051
```

```
00052 #ifdef _WIN32
00053 struct tm *http_gmtime_r(const time_t *clock, struct tm *result);
00054 #else
00055         #define http_gmtime_r gmtime_r
00056 #endif
00057
00058 int http_CancelHttpGet(/* IN */ void *Handle);
00059
00067 int http_FixUrl(
00069         uri_type *url,
00071         uri_type *fixed_url);
00072
00080 int http_FixStrUrl(
00082         const char *urlstr,
00084         size_t urlstrlen,
00086         uri_type *fixed_url);
00087
00096 SOCKET http_Connect(
00098         uri_type *destination_url,
00100         uri_type *url);
00101
00102 /************************************************************************
00103  * Function: http_RecvMessage
00104  *
00105  * Parameters:
00106  *      IN SOCKINFO *info;                      Socket information object
00107  *      OUT http_parser_t* parser;              HTTP parser object
00108  *      IN http_method_t request_method;        HTTP request method
00109  *      IN OUT int* timeout_secs;               time out
00110  *      OUT int* http_error_code;               HTTP error code returned
00111  *
00112  * Description:
00113  *      Get the data on the socket and take actions based on the read data
00114  *      to modify the parser objects buffer. If an error is reported while
00115  *      parsing the data, the error code is passed in the http_errr_code
00116  *      parameter
00117  *
00118  * Returns:
00119  *       UPNP_E_BAD_HTTPMSG
00120  *       UPNP_E_SUCCESS
00121  ************************************************************************/
00122 int http_RecvMessage(SOCKINFO *info,
00123         http_parser_t *parser,
00124         http_method_t request_method,
00125         int *timeout_secs,
00126         int *http_error_code);
00127
00150 int http_SendMessage(
00151         /* [in] Socket information object. */
00152         SOCKINFO *info,
00153         /* [in,out] Time out value. */
00154         int *timeout_secs,
00155         /* [in] Pattern format to take actions upon. */
00156         const char *fmt,
00157         /* [in] Variable parameter list. */
00158         ...);
00159
00160 /************************************************************************
00161  * Function: http_RequestAndResponse
00162  *
00163  * Parameters:
00164  *      IN uri_type* destination;       Destination URI object which contains
00165  *                                      remote IP address among other elements
00166  *      IN const char* request;         Request to be sent
00167  *      IN size_t request_length;       Length of the request
00168  *      IN http_method_t req_method;    HTTP Request method
00169  *      IN int timeout_secs;            time out value
00170  *      OUT http_parser_t* response;    Parser object to receive the repsonse
00171  *
00172  * Description:
00173  *      Initiates socket, connects to the destination, sends a
00174  *      request and waits for the response from the remote end
00175  *
00176  * Returns:
00177  *      UPNP_E_SOCKET_ERROR
00178  *      UPNP_E_SOCKET_CONNECT
00179  *      Error Codes returned by http_SendMessage
00180  *      Error Codes returned by http_RecvMessage
00181  ************************************************************************/
00182 int http_RequestAndResponse(uri_type *destination,
00183         const char *request,
00184         size_t request_length,
00185         http_method_t req_method,
00186         int timeout_secs,
00187         http_parser_t *response);
00188
00189 /************************************************************************
```

```
00190  * return codes:
00191  *      0 -- success
00192  *      UPNP_E_OUTOF_MEMORY
00193  *      UPNP_E_TIMEDOUT
00194  *      UPNP_E_BAD_REQUEST
00195  *      UPNP_E_BAD_RESPONSE
00196  *      UPNP_E_INVALID_URL
00197  *      UPNP_E_SOCKET_READ
00198  *      UPNP_E_SOCKET_WRITE
00199  ****************************************************************************/
00200
00201 /****************************************************************************
00202  * Function: http_Download
00203  *
00204  * Parameters:
00205  *      IN const char* url_str; String as a URL
00206  *      IN int timeout_secs;    time out value
00207  *      OUT char** document;    buffer to store the document extracted
00208  *                              from the donloaded message.
00209  *      OUT size_t* doc_length; length of the extracted document
00210  *      OUT char* content_type; Type of content
00211  *
00212  * Description:
00213  *      Download the document message and extract the document
00214  *      from the message.
00215  *
00216  * Return: int
00217  *      UPNP_E_SUCCESS
00218  *      UPNP_E_INVALID_URL
00219  ****************************************************************************/
00220 int http_Download(const char *url,
00221         int timeout_secs,
00222         char **document,
00223         size_t *doc_length,
00224         char *content_type);
00225
00226 /****************************************************************************
00227  * Function: http_HttpGetProgress
00228  *
00229  * Parameters:
00230  *      IN void *Handle;        Handle to the HTTP get object
00231  *      OUT size_t *length;     Buffer to get the read and parsed data
00232  *      OUT size_t *total;      Size of tge buffer passed
00233  *
00234  * Description:
00235  *      Extracts information from the Handle to the HTTP get object.
00236  *
00237  * Return: int
00238  *      UPNP_E_SUCCESS          - On Sucess
00239  *      UPNP_E_INVALID_PARAM    - Invalid Parameter
00240  ****************************************************************************/
00241 int http_HttpGetProgress(void *Handle, size_t *length, size_t *total);
00242
00266 UPNP_EXPORT_SPEC int http_OpenHttpConnection(
00269         const char *url,
00273         void **handle,
00277         int timeout);
00278
00304 UPNP_EXPORT_SPEC int http_MakeHttpRequest(
00305         /* ![in] The method to use to make the request. */
00306         Upnp_HttpMethod method,
00309         const char *url,
00311         void *handle,
00316         UpnpString *headers,
00318         const char *contentType,
00323         int contentLength,
00327         int timeout);
00328
00343 UPNP_EXPORT_SPEC int http_WriteHttpRequest(
00346         void *handle,
00348         char *buf,
00350         size_t *size,
00354         int timeout);
00355
00374 UPNP_EXPORT_SPEC int http_EndHttpRequest(
00376         void *handle,
00380         int timeout);
00381
00411 UPNP_EXPORT_SPEC int http_GetHttpResponse(
00414         void *handle,
00417         UpnpString *headers,
00419         char **contentType,
00421         int *contentLength,
00423         int *httpStatus,
00428         int timeout);
00429
00448 UPNP_EXPORT_SPEC int http_ReadHttpResponse(
```

```
00451         void *handle,
00453         char *buf,
00455         size_t *size,
00460         int timeout);
00461
00474 UPNP_EXPORT_SPEC int http_CloseHttpConnection(
00477         void *handle);
00478
00479 /************************************************************************
00480  * Function: http_SendStatusResponse
00481  *
00482  * Parameters:
00483  *      IN SOCKINFO *info;              Socket information object
00484  *      IN int http_status_code;        error code returned while making
00485  *                                      or sending the response message
00486  *      IN int request_major_version;   request major version
00487  *      IN int request_minor_version;   request minor version
00488  *
00489  * Description:
00490  *      Generate a response message for the status query and send the
00491  *      status response.
00492  *
00493  * Return: int
00494  *      0 -- success
00495  *      UPNP_E_OUTOF_MEMORY
00496  *      UPNP_E_SOCKET_WRITE
00497  *      UPNP_E_TIMEDOUT
00498  ************************************************************************/
00499 int http_SendStatusResponse(SOCKINFO *info,
00500         int http_status_code,
00501         int request_major_version,
00502         int request_minor_version);
00503
00544 int http_MakeMessage(
00545         /* [in,out] Buffer with the contents of the message. */
00546         membuffer *buf,
00547         /* [in] HTTP major version. */
00548         int http_major_version,
00549         /* [in] HTTP minor version. */
00550         int http_minor_version,
00551         /* [in] Pattern format. */
00552         const char *fmt,
00553         /* [in] Format arguments. */
00554         ...);
00555
00556 /************************************************************************
00557  * Function: http_CalcResponseVersion
00558  *
00559  * Parameters:
00560  *      IN int request_major_vers;      Request major version
00561  *      IN int request_minor_vers;      Request minor version
00562  *      OUT int* response_major_vers;   Response mojor version
00563  *      OUT int* response_minor_vers;   Response minor version
00564  *
00565  * Description:
00566  *      Calculate HTTP response versions based on the request versions.
00567  *
00568  * Return: void
00569  ************************************************************************/
00570 void http_CalcResponseVersion(int request_major_vers,
00571         int request_minor_vers,
00572         int *response_major_vers,
00573         int *response_minor_vers);
00574
00575 /************************************************************************
00576  * Function: http_OpenHttpGetEx
00577  *
00578  * Parameters:
00579  *      IN const char *url_str;         String as a URL
00580  *      IN OUT void **Handle;           Pointer to buffer to store HTTP
00581  *                                      post handle
00582  *      IN OUT char **contentType;      Type of content
00583  *      OUT int *contentLength;         length of content
00584  *      OUT int *httpStatus;            HTTP status returned on receiving a
00585  *                                      response message
00586  *      IN int timeout;                 time out value
00587  *
00588  * Description:
00589  *      Makes the HTTP GET message, connects to the peer,
00590  *      sends the HTTP GET request, gets the response and parses the
00591  *      response.
00592  *
00593  * Return: int
00594  *      UPNP_E_SUCCESS          - On Success
00595  *      UPNP_E_INVALID_PARAM    - Invalid Paramters
00596  *      UPNP_E_OUTOF_MEMORY
00597  *      UPNP_E_SOCKET_ERROR
```

```
00598  *      UPNP_E_BAD_RESPONSE
00599  ***************************************************************************/
00600 int http_OpenHttpGetEx(const char *url_str,
00601         void **Handle,
00602         char **contentType,
00603         int *contentLength,
00604         int *httpStatus,
00605         int lowRange,
00606         int highRange,
00607         int timeout);
00608
00609 /************************************************************************
00610  * Function: get_sdk_info
00611  *
00612  * Parameters:
00613  *      OUT char *info; buffer to store the operating system information
00614  *      IN size_t infoSize; size of buffer
00615  *
00616  * Description:
00617  *      Returns the server information for the operating system
00618  *
00619  * Return:
00620  *      UPNP_INLINE void
00621  ************************************************************************/
00622 void get_sdk_info(char *info, size_t infoSize);
00623
00624 #ifdef __cplusplus
00625 } /* #extern "C" */
00626 #endif
00627
00628 #endif /* GENLIB_NET_HTTP_HTTPREADWRITE_H */
```

9.111 inet_pton.h

```
00001 #ifndef INET_PTON
00002 #define INET_PTON
00003
00004 #ifdef _WIN32
00005
00006         #ifdef IPV6_
00007                 #define INET_IPV6
00008         #endif
00009
00010         #include "unixutil.h"
00011
00012         #include <errno.h>
00013         #include <stdio.h>
00014         #include <string.h>
00015
00016         #if (NTDDI_VERSION < NTDDI_VISTA)
00017
00036 extern const char *inet_ntop(
00037         int af, const void *src, char *dst, socklen_t size);
00038
00049 extern int inet_pton(int af, const char *src, void *dst);
00050
00051         #endif /* NTDDI_VERSION < NTDDI_VISTA */
00052
00053 #endif /* _WIN32 */
00054
00055 #endif /* INET_PTON */
```

9.112 md5.h

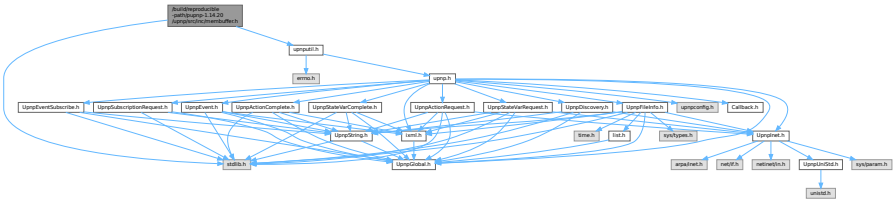
```
00001 /*      $OpenBSD: md5.h,v 1.3 2014/11/16 17:39:09 tedu Exp $    */
00002
00003 /*
00004  * This code implements the MD5 message-digest algorithm.
00005  * The algorithm is due to Ron Rivest.  This code was
00006  * written by Colin Plumb in 1993, no copyright is claimed.
00007  * This code is in the public domain; do with it what you wish.
00008  *
00009  * Equivalent code is available from RSA Data Security, Inc.
00010  * This code has been tested against that, and is equivalent,
00011  * except that you don't need to include two pages of legalese
00012  * with every copy.
00013  */
00014
00015 #ifndef _MD5_H_
```

```
00016 #define _MD5_H_
00017
00018 #include "UpnpStdInt.h"
00019 #include <stddef.h>
00020
00021 #define MD5_BLOCK_LENGTH 64
00022 #define MD5_DIGEST_LENGTH 16
00023
00024 typedef struct MD5Context
00025 {
00026         uint32_t state[4];                /* state */
00027         uint64_t count;                   /* number of bits, mod 2^64 */
00028         uint8_t buffer[MD5_BLOCK_LENGTH]; /* input buffer */
00029 } MD5_CTX;
00030
00031 void MD5Init(MD5_CTX *);
00032 void MD5Update(MD5_CTX *, const void *, size_t);
00033 void MD5Final(uint8_t[MD5_DIGEST_LENGTH], MD5_CTX *);
00034 void MD5Transform(uint32_t[4], const uint8_t[MD5_BLOCK_LENGTH]);
00035
00036 #endif /* _MD5_H_ */
```

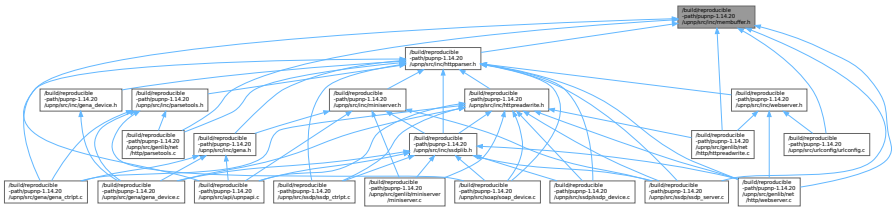
9.113 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/membuffer.h File Reference

```
#include "upnputil.h"
#include <stdlib.h>
```

Include dependency graph for membuffer.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct memptr
- struct membuffer

Macros

- #define **MINVAL**(a, b) ((a) < (b) ? (a) : (b))
- #define **MAXVAL**(a, b) ((a) > (b) ? (a) : (b))
- #define MEMBUF_DEF_SIZE_INC (size_t)5

Functions

- char * str_alloc (const char *str, size_t str_len)

  Allocate memory and copy information from the input string to the newly allocated memory.
- int memptr_cmp (memptr *m, const char *s)

  Compares characters of strings passed for number of bytes. If equal for the number of bytes, the length of the bytes determines which buffer is shorter.
- int memptr_cmp_nocase (memptr *m, const char *s)

  Compares characters of 2 strings irrespective of the case for a specific count of bytes.
- int membuffer_set_size (membuffer *m, size_t new_length)

  Increases or decreases buffer cap so that at least 'new_length' bytes can be stored.
- void membuffer_init (membuffer *m)

  Wrapper to membuffer_initialize().
- void membuffer_destroy (membuffer *m)

  Free's memory allocated for membuffer m.*
- int membuffer_assign (membuffer *m, const void *buf, size_t buf_len)

  *Allocate memory to membuffer *m and copy the contents of the in parameter const void *buf.*
- int membuffer_assign_str (membuffer *m, const char *c_str)

  Wrapper function for membuffer_assign().
- int membuffer_append (membuffer *m, const void *buf, size_t buf_len)

  Invokes function to appends data from a constant buffer to the buffer.
- int membuffer_append_str (membuffer *m, const char *c_str)

  Invokes function to appends data from a constant string to the buffer.
- int membuffer_insert (membuffer *m, const void *buf, size_t buf_len, size_t index)

  Allocates memory for the new data to be inserted. Does memory management by moving the data from the existing memory to the newly allocated memory and then appending the new data.
- void membuffer_delete (membuffer *m, size_t index, size_t num_bytes)

  Shrink the size of the buffer depending on the current size of the bufer and te input parameters. Move contents from the old buffer to the new sized buffer.
- char * membuffer_detach (membuffer *m)
- void membuffer_attach (membuffer *m, char *new_buf, size_t buf_len)

9.113.1 Macro Definition Documentation

9.113.1.1 MEMBUF_DEF_SIZE_INC

```
#define MEMBUF_DEF_SIZE_INC (size_t)5
```

default value of size_inc.

9.113.2 Function Documentation

9.113.2.1 membuffer_append()

```
int membuffer_append (
            membuffer * m,
            const void * buf,
            size_t buf_len )
```

Invokes function to appends data from a constant buffer to the buffer.

Returns

int.

Parameters

in,out	*m*	Buffer whose memory is to be appended.
in	*buf*	Source buffer whose contents will be copied.
in	*buf_len*	Length of the source buffer.

References membuffer::length.

9.113.2.2 membuffer_append_str()

```
int membuffer_append_str (
            membuffer * m,
            const char * c_str )
```

Invokes function to appends data from a constant string to the buffer.

Returns

int.

Parameters

in,out	*m*	Buffer whose memory is to be appended.
in	*c_str*	Source buffer whose contents will be copied.

References membuffer::length.

9.113.2.3 membuffer_assign()

```
int membuffer_assign (
            membuffer * m,
            const void * buf,
            size_t buf_len )
```

Allocate memory to membuffer ∗m and copy the contents of the in parameter const void ∗buf.

Returns

- UPNP_E_SUCCESS
- UPNP_E_OUTOF_MEMORY

Parameters

in,out	*m*	Buffer whose memory is to be allocated and assigned.
in	*buf*	Source buffer whose contents will be copied.
in	*buf_len*	Length of the source buffer.

References membuffer::buf, and membuffer::length.

9.113.2.4 membuffer_assign_str()

```
int membuffer_assign_str (
            membuffer * m,
            const char * c_str )
```

Wrapper function for membuffer_assign().

Returns

- UPNP_E_SUCCESS
- UPNP_E_OUTOF_MEMORY

Parameters

in,out	*m*	Buffer to be allocated and assigned.
in	*c_str*	Source buffer whose contents will be copied.

9.113.2.5 membuffer_attach()

```
void membuffer_attach (
            membuffer * m,
            char * new_buf,
            size_t buf_len )
```

Parameters

in,out	*m*	Buffer to be updated.
in	*new_buf*	Source buffer which will be assigned to the buffer to be updated.
in	*buf_len*	Length of the source buffer.

References membuffer::buf, membuffer::capacity, and membuffer::length.

9.113.2.6 membuffer_delete()

```
void membuffer_delete (
            membuffer * m,
            size_t index,
            size_t num_bytes )
```

Shrink the size of the buffer depending on the current size of the bufer and te input parameters. Move contents from the old buffer to the new sized buffer.

Parameters

in,out	*m*	Buffer whose memory size is to be decreased and copied to the modified location.
in	*index*	Index to determine bounds while moving data.
in	*num_bytes*	Number of bytes that the data needs to shrink by.

References membuffer::buf, and membuffer::length.

9.113.2.7 membuffer_destroy()

```
void membuffer_destroy (
            membuffer * m )
```

Free's memory allocated for membuffer* m.

Parameters

in,out	*m*	Buffer to be destroyed.

References membuffer::buf.

9.113.2.8 membuffer_detach()

```
char * membuffer_detach (
            membuffer * m )
```

Parameters

in,out	*m*	Buffer to be returned and updated.

References membuffer::buf.

9.113.2.9 membuffer_init()

```
void membuffer_init (
            membuffer * m )
```

Wrapper to membuffer_initialize().

Set the size of the buffer to MEMBUF_DEF_SIZE_INC and Initializes m->buf to NULL, length = 0.

Parameters

in,out	*m*	Buffer to be initialized.

References MEMBUF_DEF_SIZE_INC, and membuffer::size_inc.

9.113.2.10 membuffer_insert()

```
int membuffer_insert (
            membuffer * m,
            const void * buf,
```

```
            size_t buf_len,
            size_t index )
```

Allocates memory for the new data to be inserted. Does memory management by moving the data from the existing memory to the newly allocated memory and then appending the new data.

Returns

0 if successful, error code if error.

Parameters

in,out	*m*	Buffer whose memory size is to be increased and appended.
in	*buf*	source buffer whose contents will be copied.
in	*buf_len*	size of the source buffer.
in	*index*	index to determine the bounds while movinf the data.

References membuffer::buf, and membuffer::length.

9.113.2.11 membuffer_set_size()

```
int membuffer_set_size (
            membuffer * m,
            size_t new_length )
```

Increases or decreases buffer cap so that at least 'new_length' bytes can be stored.

Returns

- UPNP_E_SUCCESS - On Success
- UPNP_E_OUTOF_MEMORY - On failure to allocate memory.

Parameters

in,out	*m*	buffer whose size is to be modified.
in	*new_length*	new size to which the buffer will be modified.

References membuffer::buf, membuffer::capacity, membuffer::length, membuffer::size_inc, and UPNP_E_OUTOF_MEMORY.

9.113.2.12 memptr_cmp()

```
int memptr_cmp (
            memptr * m,
            const char * s )
```

Compares characters of strings passed for number of bytes. If equal for the number of bytes, the length of the bytes determines which buffer is shorter.

Returns

- < 0 string1 substring less than string2 substring
- == 0 string1 substring identical to string2 substring
- > 0 string1 substring greater than string2 substring

Parameters

in	*m*	Input memory object.
in	*s*	Constatnt string for the memory object to be compared with.

References memptr::buf, and memptr::length.

9.113.2.13 memptr_cmp_nocase()

```
int memptr_cmp_nocase (
            memptr * m,
            const char * s )
```

Compares characters of 2 strings irrespective of the case for a specific count of bytes.

If the character comparison is the same the length of the 2 srings determines the shorter of the 2 strings.

Returns

- < 0 string1 substring less than string2 substring
- == 0 string1 substring identical to string2 substring
- > 0 string1 substring greater than string2 substring

Parameters

in	*m*	Input memory object.
in	*s*	Constatnt string for the memory object to be compared with.

References memptr::buf, and memptr::length.

9.113.2.14 str_alloc()

```
char * str_alloc (
            const char * str,
            size_t str_len )
```

Allocate memory and copy information from the input string to the newly allocated memory.

Returns

Pointer to the newly allocated memory. NULL if memory cannot be allocated.

Parameters

in	*str*	Input string object.
in	*str_len*	Input string length.

9.114 membuffer.h

Go to the documentation of this file.

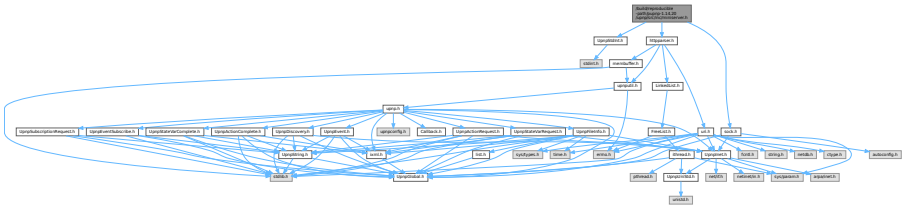
```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (c) 2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * - Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * - Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * - Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef GENLIB_UTIL_MEMBUFFER_H
00034 #define GENLIB_UTIL_MEMBUFFER_H
00035
00040 #include "upnputil.h"
00041 #include <stdlib.h>
00042
00043 #define MINVAL(a, b) ((a) < (b) ? (a) : (b))
00044 #define MAXVAL(a, b) ((a) > (b) ? (a) : (b))
00045
00047 typedef struct
00048 {
00050         char *buf;
00052         size_t length;
00053 } memptr;
00054
00057 typedef struct
00058 {
00060         char *buf;
00062         size_t length;
00064         size_t capacity;
00066         size_t size_inc;
00068 #define MEMBUF_DEF_SIZE_INC (size_t)5
00069 } membuffer;
00070
00071 #ifdef __cplusplus
00072 extern "C" {
00073 #endif /* __cplusplus */
00074
00082 char *str_alloc(
00084         const char *str,
00086         size_t str_len);
00087
00098 int memptr_cmp(
00100         memptr *m,
00102         const char *s);
00103
00116 int memptr_cmp_nocase(
```

```
00118         memptr *m,
00120         const char *s);
00121
00130 int membuffer_set_size(
00132         membuffer *m,
00134         size_t new_length);
00135
00142 void membuffer_init(
00144         membuffer *m);
00145
00149 void membuffer_destroy(
00151         membuffer *m);
00152
00161 int membuffer_assign(
00163         membuffer *m,
00165         const void *buf,
00167         size_t buf_len);
00168
00176 int membuffer_assign_str(
00178         membuffer *m,
00180         const char *c_str);
00181
00187 int membuffer_append(
00189         membuffer *m,
00191         const void *buf,
00193         size_t buf_len);
00194
00200 int membuffer_append_str(
00202         membuffer *m,
00204         const char *c_str);
00205
00213 int membuffer_insert(
00216         membuffer *m,
00218         const void *buf,
00220         size_t buf_len,
00222         size_t index);
00223
00229 void membuffer_delete(
00232         membuffer *m,
00234         size_t index,
00236         size_t num_bytes);
00237
00238 /*
00239  * \brief Detaches current buffer and returns it. The caller must free the
00240  * returned buffer using free(). After this call, length becomes 0.
00241  *
00242  * \return A pointer to the current buffer.
00243  */
00244 char *membuffer_detach(
00246         membuffer *m);
00247
00248 /*
00249  * \brief Free existing memory in membuffer and assign the new buffer in its
00250  * place.
00251  *
00252  * \note 'new_buf' must be allocted using malloc or realloc so that it can be
00253  * freed using free().
00254  */
00255 void membuffer_attach(
00257         membuffer *m,
00260         char *new_buf,
00262         size_t buf_len);
00263
00264 #ifdef __cplusplus
00265 } /* extern "C" */
00266 #endif /* __cplusplus */
00267
00268 #endif /* GENLIB_UTIL_MEMBUFFER_H */
```

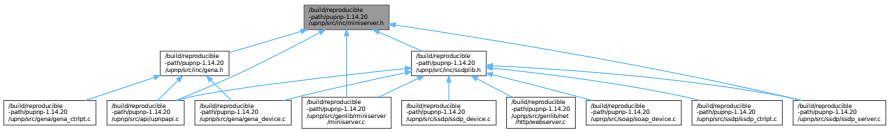
9.115 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/miniserver.h File Reference

```
#include "UpnpStdInt.h"
#include "httpparser.h"
#include "sock.h"
```

Include dependency graph for miniserver.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct MServerSockArray

Typedefs

- typedef struct MServerSockArray **MiniServerSockArray**
- typedef void(∗ MiniServerCallback) (http_parser_t ∗parser, http_message_t ∗request, SOCKINFO ∗info)

Functions

- void SetHTTPGetCallback (MiniServerCallback callback)

  Set HTTP Get Callback.
- void SetSoapCallback (MiniServerCallback callback)

  Set SOAP Callback.
- void SetGenaCallback (MiniServerCallback callback)

  Set GENA Callback.
- int StartMiniServer (uint16_t ∗listen_port4, uint16_t ∗listen_port6, uint16_t ∗listen_port6UlaGua)

  Initialize the sockets functionality for the Miniserver.
- int StopMiniServer ()

  Stop and Shutdown the MiniServer and free socket resources.

Variables

- SOCKET **gMiniServerStopSock**

9.115.1 Typedef Documentation

9.115.1.1 MiniServerCallback

```
typedef void(* MiniServerCallback) ( http_parser_t *parser, http_message_t *request, SOCKINFO
*info)
```

9.115.2 Function Documentation

9.115.2.1 SetGenaCallback()

```
void SetGenaCallback (
            MiniServerCallback callback )
```

Set GENA Callback.

Parameters

in	*callback*	GENA Callback to be invoked.

Referenced by UpnpInitPreamble().

9.115.2.2 SetHTTPGetCallback()

```
void SetHTTPGetCallback (
            MiniServerCallback callback )
```

Set HTTP Get Callback.

Parameters

in	*callback*	HTTP Callback to be invoked .

Referenced by UpnpEnableWebserver().

9.115.2.3 SetSoapCallback()

```
void SetSoapCallback (
            MiniServerCallback callback )
```

Set SOAP Callback.

Parameters

in	*callback*	SOAP Callback to be invoked .

Referenced by UpnpInitPreamble().

9.115.2.4 StartMiniServer()

```
int StartMiniServer (
            uint16_t * listen_port4,
            uint16_t * listen_port6,
            uint16_t * listen_port6UlaGua )
```

Initialize the sockets functionality for the Miniserver.

Initialize a thread pool job to run the MiniServer and the job to the thread pool.

If listen port is 0, port is dynamically picked.

Use timer mechanism to start the MiniServer, failure to meet the allowed delay aborts the attempt to launch the MiniServer.

Returns

- On success: UPNP_E_SUCCESS.
- On error: UPNP_E_XXX.

Parameters

in,out	*listen_port4*	Port on which the server listens for incoming IPv4 connections.
in,out	*listen_port6*	Port on which the server listens for incoming IPv6 LLA connections.
in,out	*listen_port6UlaGua*	Port on which the server listens for incoming IPv6 ULA or GUA connections.

References get_miniserver_stopsock(), get_ssdp_sockets(), gMiniServerThreadPool, gMServState, MServerSockArray::miniServerSo MServerSockArray::miniServerSock6, MServerSockArray::miniServerSock6UlaGua, MServerSockArray::miniServerStopSock, MSERV_IDLE, MSERV_RUNNING, RunMiniServer(), sock_close(), MServerSockArray::ssdpReqSock4, MServerSockArray::ssdpReqSock6, MServerSockArray::ssdpSock4, MServerSockArray::ssdpSock6, MServerSockArray::ssdpSock6 ThreadPoolAddPersistent(), TPJobInit(), TPJobSetFreeFunction(), TPJobSetPriority(), UPNP_E_INTERNAL_ERROR, UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by UpnpInitStartServers().

9.115.2.5 StopMiniServer()

```
int StopMiniServer ( )
```

Stop and Shutdown the MiniServer and free socket resources.

Returns

Always returns 0.

References gMServState, INVALID_SOCKET, miniStopSockPort, MSERV_IDLE, MSERV_RUNNING, MSERV_STOPPING, sock_close(), and UpnpPrintf().

Referenced by UpnpFinish().

9.116 miniserver.h

Go to the documentation of this file.

```
00001 #ifndef MINISERVER_H
00002 #define MINISERVER_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  *
00009  * Redistribution and use in source and binary forms, with or without
00010  * modification, are permitted provided that the following conditions are met:
00011  *
00012  * - Redistributions of source code must retain the above copyright notice,
00013  * this list of conditions and the following disclaimer.
00014  * - Redistributions in binary form must reproduce the above copyright notice,
00015  * this list of conditions and the following disclaimer in the documentation
00016  * and/or other materials provided with the distribution.
00017  * - Neither name of Intel Corporation nor the names of its contributors
00018  * may be used to endorse or promote products derived from this software
00019  * without specific prior written permission.
00020  *
00021  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00022  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00023  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00024  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00025  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00026  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00027  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00028  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00029  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00030  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00031  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00032  *
00033  ******************************************************************************/
00034
00039 #include "UpnpStdInt.h"
00040 #include "httpparser.h"
00041 #include "sock.h"
00042
00043 extern SOCKET gMiniServerStopSock;
00044
00045 typedef struct MServerSockArray
00046 {
00048         SOCKET miniServerSock4;
00050         SOCKET miniServerSock6;
00052         SOCKET miniServerSock6UlaGua;
00054         SOCKET miniServerStopSock;
00056         SOCKET ssdpSock4;
00059         SOCKET ssdpSock6;
00062         SOCKET ssdpSock6UlaGua;
00063         /* ! . */
00064         uint16_t stopPort;
00065         /* ! . */
00066         uint16_t miniServerPort4;
00067         /* ! . */
00068         uint16_t miniServerPort6;
00069         /* ! . */
00070         uint16_t miniServerPort6UlaGua;
00071 #ifdef INCLUDE_CLIENT_APIS
00074         SOCKET ssdpReqSock4;
00077         SOCKET ssdpReqSock6;
00078 #endif /* INCLUDE_CLIENT_APIS */
00079 } MiniServerSockArray;
00080
00082 typedef void (*MiniServerCallback)(
00083         /* ! [in] . */
00084         http_parser_t *parser,
00085         /* ! [in] . */
00086         http_message_t *request,
00087         /* ! [in] . */
00088         SOCKINFO *info);
00089
00090 #ifdef __cplusplus
00091 extern "C" {
00092 #endif
00093
00097 void SetHTTPGetCallback(
00099         MiniServerCallback callback);
00100
00104 #ifdef INCLUDE_DEVICE_APIS
00105 void SetSoapCallback(
00107         MiniServerCallback callback);
00108 #else  /* INCLUDE_DEVICE_APIS */
```

```
00109 static UPNP_INLINE void SetSoapCallback(MiniServerCallback callback) {}
00110 #endif /* INCLUDE_DEVICE_APIS */
00114 void SetGenaCallback(
00116         MiniServerCallback callback);
00117
00133 int StartMiniServer(
00136         uint16_t *listen_port4,
00139         uint16_t *listen_port6,
00142         uint16_t *listen_port6UlaGua);
00143
00149 int StopMiniServer();
00150
00151 #ifdef __cplusplus
00152 } /* extern C */
00153 #endif
00154
00155 #endif /* MINISERVER_H */
```

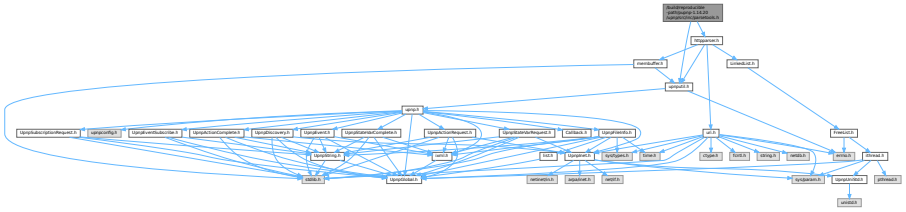
9.117 netall.h

```
00001
00002 //
00003 // Copyright (c) 2000-2003 Intel Corporation
00004 // All rights reserved.
00005 //
00006 // Redistribution and use in source and binary forms, with or without
00007 // modification, are permitted provided that the following conditions are met:
00008 //
00009 // * Redistributions of source code must retain the above copyright notice,
00010 // this list of conditions and the following disclaimer.
00011 // * Redistributions in binary form must reproduce the above copyright notice,
00012 // this list of conditions and the following disclaimer in the documentation
00013 // and/or other materials provided with the distribution.
00014 // * Neither name of Intel Corporation nor the names of its contributors
00015 // may be used to endorse or promote products derived from this software
00016 // without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023 // EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024 // PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025 // PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026 // OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027 // NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028 // SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00031
00032 #include <genlib/net/netexception.h>
00033 #include <genlib/net/netreader.h>
```

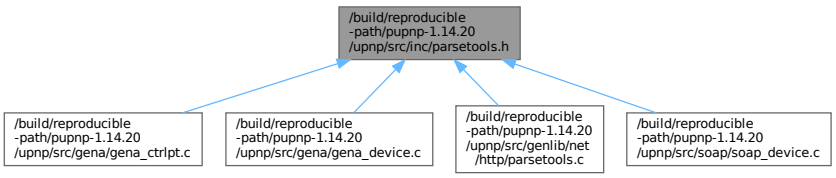
9.118 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/parsetools.h File Reference

```
#include "httpparser.h"
#include "upnputil.h"
```

Include dependency graph for parsetools.h:



This graph shows which files directly or indirectly include this file:



Functions

- int has_xml_content_type (http_message_t *hmsg)

  Find the header from the HTTP message and match the header for xml data.

9.118.1 Function Documentation

9.118.1.1 has_xml_content_type()

```
int has_xml_content_type (
            http_message_t * hmsg )
```

Find the header from the HTTP message and match the header for xml data.

Returns

boolean.

Parameters

in	*hmsg*	HTTP Message object.

References memptr::buf, memptr::length, and PARSE_OK.

Referenced by gena_process_notification_event(), and soap_device_callback().

9.119 parsetools.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
```

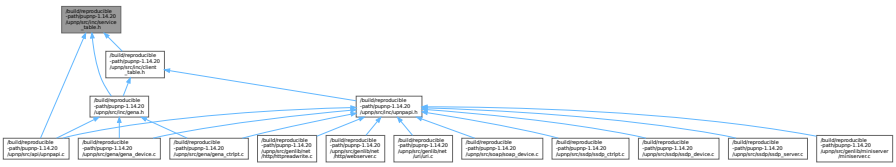
```
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENLIB_NET_HTTP_PARSETOOLS_H
00033 #define GENLIB_NET_HTTP_PARSETOOLS_H
00034
00039 #include "httpparser.h"
00040 #include "upnputil.h"
00041
00042 #ifdef __cplusplus
00043 extern "C" {
00044 #endif
00045
00052 int has_xml_content_type(
00054         http_message_t *hmsg);
00055
00056 #ifdef __cplusplus
00057 } /* extern C */
00058 #endif
00059
00060 #endif /* GENLIB_NET_HTTP_PARSETOOLS_H */
```

9.120 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/service_table.h File Reference

```
#include "LinkedList.h"
#include "config.h"
#include "ixml.h"
#include "upnp.h"
#include "upnpdebug.h"
#include "uri.h"
#include <stdio.h>
#include <time.h>
```

Include dependency graph for service_table.h:

This graph shows which files directly or indirectly include this file:



Data Structures

- struct SUBSCRIPTION
- struct SERVICE_INFO
- struct SERVICE_TABLE

Macros

- #define **SID_SIZE** (size_t)41

Typedefs

- typedef struct SUBSCRIPTION **subscription**
- typedef struct SERVICE_INFO **service_info**
- typedef struct SERVICE_TABLE **service_table**

Functions

- void **freeSubscriptionQueuedEvents** (subscription *sub)
- int copy_subscription (subscription *in, subscription *out)

  Makes a copy of the subscription.
- void RemoveSubscriptionSID (Upnp_SID sid, service_info *service)
- subscription * GetSubscriptionSID (const Upnp_SID sid, service_info *service)

  Return the subscription from the service table that matches const Upnp_SID sid value.
- subscription * GetFirstSubscription (service_info *service)

  Gets pointer to the first subscription node in the service table.
- subscription * GetNextSubscription (service_info *service, subscription *current)

  Get current and valid subscription from the service table.
- void freeSubscription (subscription *sub)

  Free's the memory allocated for storing the URL of the subscription.
- void freeSubscriptionList (subscription *head)

  Free's memory allocated for all the subscriptions in the service table.
- service_info * FindServiceId (service_table *table, const char *serviceId, const char *UDN)

  Traverses through the service table and returns a pointer to the service node that matches a known service id and a known UDN.
- service_info * FindServiceEventURLPath (service_table *table, const char *eventURLPath)

  Traverses the service table and finds the node whose event URL Path matches a know value.
- service_info * FindServiceControlURLPath (service_table *table, const char *controlURLPath)

  Traverses the service table and finds the node whose control URL Path matches a know value.
- void printService (service_info *service, Upnp_LogLevel level, Dbg_Module module)

  For debugging purposes prints information from the service passed into the function.
- void printServiceList (service_info *service, Upnp_LogLevel level, Dbg_Module module)

  For debugging purposes prints information of each service from the service table passed into the function.
- void printServiceTable (service_table *table, Upnp_LogLevel level, Dbg_Module module)

  For debugging purposes prints the URL base of the table and information of each service from the service table passed into the function.
- void freeService (service_info *in)

  Free's memory allocated for the various components of the service entry in the service table.
- void freeServiceList (service_info *head)

  Free's memory allocated for the various components of each service entry in the service table.

- void freeServiceTable (service_table *table)

  Free's dynamic memory in table (does not free table, only memory within the structure).
- int removeServiceTable (IXML_Node *node, service_table *in)

  This function assumes that services for a particular root device are placed linearly in the service table, and in the order in which they are found in the description document all services for this root device are removed from the list.
- int addServiceTable (IXML_Node *node, service_table *in, const char *DefaultURLBase)

  Add Service to the table.
- int getServiceTable (IXML_Node *node, service_table *out, const char *DefaultURLBase)

  Retrieve service from the table.
- DOMString getElementValue (IXML_Node *node)

  Returns the clone of the element value.
- int getSubElement (const char *element_name, IXML_Node *node, IXML_Node **out)

  Traverses through a list of XML nodes to find the node with the known element name.

9.120.1 Function Documentation

9.120.1.1 addServiceTable()

```
int addServiceTable (
            IXML_Node * node,
            service_table * in,
            const char * DefaultURLBase )
```

Add Service to the table.

Parameters

in	*node*	XML node information.
in	*in*	Service table that will be initialized with services.
in	*DefaultURLBase*	Default base URL on which the URL will be returned to the service list.

References ixmlCloneDOMString().

9.120.1.2 copy_subscription()

```
int copy_subscription (
            subscription * in,
            subscription * out )
```

Makes a copy of the subscription.

Returns

HTTP_SUCCESS on success.

Parameters

in	*in*	Source subscription.
in	*out*	Destination subscription.

References copy_URL_list().

9.120.1.3 FindServiceControlURLPath()

```
service_info * FindServiceControlURLPath (
            service_table * table,
            const char * controlURLPath )
```

Traverses the service table and finds the node whose control URL Path matches a know value.

Returns

Pointer to the service list node from the service table whose control URL Path matches a known value.

Parameters

in	*table*	Service table.
in	*controlURLPath*	Control URL path used to find a service from the table.

References parse_uri(), and token_cmp().

9.120.1.4 FindServiceEventURLPath()

```
service_info * FindServiceEventURLPath (
            service_table * table,
            const char * eventURLPath )
```

Traverses the service table and finds the node whose event URL Path matches a know value.

Returns

Pointer to the service list node from the service table whose event URL matches a known event URL.

Parameters

in	*table*	Service table.
in	*eventURLPath*	Event URL path used to find a service from the table.

References parse_uri(), and token_cmp().

9.120.1.5 FindServiceId()

```
service_info * FindServiceId (
            service_table * table,
            const char * serviceId,
            const char * UDN )
```

Traverses through the service table and returns a pointer to the service node that matches a known service id and a known UDN.

Returns

Pointer to the matching service_info node.

Parameters

in	*table*	Service table.
in	*service↩ Id*	String representing the service id to be found among those in the table.
in	*UDN*	String representing the UDN to be found among those in the table.

9.120.1.6 freeService()

```
void freeService (
            service_info * in )
```

Free's memory allocated for the various components of the service entry in the service table.

Parameters

in	*in*	Service information that is to be freed.

References ixmlFreeDOMString().

9.120.1.7 freeServiceList()

```
void freeServiceList (
            service_info * head )
```

Free's memory allocated for the various components of each service entry in the service table.

Parameters

in	*head*	Head of the service list to be freed.

References ixmlFreeDOMString().

9.120.1.8 freeServiceTable()

```
void freeServiceTable (
            service_table * table )
```

Free's dynamic memory in table (does not free table, only memory within the structure).

Parameters

in	*table*	Service table whose internal memory needs to be freed.

References ixmlFreeDOMString().

9.120.1.9 freeSubscription()

```
void freeSubscription (
            subscription * sub )
```

Free's the memory allocated for storing the URL of the subscription.

Parameters

in	*sub*	Subscription object to be freed.

References free_URL_list().

9.120.1.10 freeSubscriptionList()

```
void freeSubscriptionList (
            subscription * head )
```

Free's memory allocated for all the subscriptions in the service table.

Parameters

in	*head*	Head of the subscription list.

9.120.1.11 getElementValue()

```
DOMString getElementValue (
            IXML_Node * node )
```

Returns the clone of the element value.

Note

Value must be freed with DOMString_free.

Returns

DOMString

Parameters

in	*node*	Input node which provides the list of child nodes.

References DOMString, ixmlCloneDOMString(), ixmlNode_getFirstChild(), ixmlNode_getNodeType(), and ixmlNode_getNodeValue().

9.120.1.12 GetFirstSubscription()

```
subscription * GetFirstSubscription (
            service_info * service )
```

Gets pointer to the first subscription node in the service table.

Returns

Pointer to the first subscription node.

Parameters

in	*service*	Service object providing the list of subscriptions.

9.120.1.13 GetNextSubscription()

```
subscription * GetNextSubscription (
            service_info * service,
            subscription * current )
```

Get current and valid subscription from the service table.

Returns

Pointer to the next subscription node.

Parameters

in	*service*	Service object providing the list of subscriptions.
in	*current*	Current subscription object.

9.120.1.14 getServiceTable()

```
int getServiceTable (
            IXML_Node * node,
            service_table * out,
            const char * DefaultURLBase )
```

Retrieve service from the table.

Returns

An integer

Parameters

in	*node*	XML node information.
in	*out*	Output parameter which will contain the service list and URL.
in	*DefaultURLBase*	Default base URL on which the URL will be returned.

References ixmlCloneDOMString().

9.120.1.15 getSubElement()

```
int getSubElement (
            const char * element_name,
            IXML_Node * node,
            IXML_Node ** out )
```

Traverses through a list of XML nodes to find the node with the known element name.

Returns

- 1 - On Success
- 0 - On Failure

Parameters

in	*element_name*	Sub element name to be searched for.
in	*node*	Input node which provides the list of child nodes.
out	*out*	Ouput node to which the matched child node is returned.

References DOMString, ixmlNode_getFirstChild(), ixmlNode_getNextSibling(), ixmlNode_getNodeName(), and ixmlNode_getNodeType().

9.120.1.16 GetSubscriptionSID()

```
subscription * GetSubscriptionSID (
            const Upnp_SID sid,
            service_info * service )
```

Return the subscription from the service table that matches const Upnp_SID sid value.

Returns

Pointer to the matching subscription node.

Parameters

in	*sid*	Subscription ID.
in	*service*	Service object providing the list of subscriptions.

9.120.1.17 printService()

```
void printService (
            service_info * service,
```

```
            Upnp_LogLevel level,
            Dbg_Module module )
```

For debugging purposes prints information from the service passed into the function.

Parameters

in	*service*	Service whose information is to be printed.
in	*level*	Debug level specified to the print function.
in	*module*	Debug module specified to the print function.

References UpnpPrintf().

9.120.1.18 printServiceList()

```
void printServiceList (
            service_info * service,
            Upnp_LogLevel level,
            Dbg_Module module )
```

For debugging purposes prints information of each service from the service table passed into the function.

Parameters

in	*service*	Service whose information is to be printed.
in	*level*	Debug level specified to the print function.
in	*module*	Debug module specified to the print function.

References UpnpPrintf().

9.120.1.19 printServiceTable()

```
void printServiceTable (
            service_table * table,
            Upnp_LogLevel level,
            Dbg_Module module )
```

For debugging purposes prints the URL base of the table and information of each service from the service table passed into the function.

Parameters

in	*table*	Service table to be printed.
in	*level*	Debug level specified to the print function.
in	*module*	Debug module specified to the print function.

References UpnpPrintf().

9.120.1.20 removeServiceTable()

```
int removeServiceTable (
            IXML_Node * node,
            service_table * in )
```

This function assumes that services for a particular root device are placed linearly in the service table, and in the order in which they are found in the description document all services for this root device are removed from the list.

Returns

An integer.

Parameters

in	*node*	XML node information.
in	*in*	Service table from which services will be removed.

References DOMString, ixmlElement_getElementsByTagName(), ixmlFreeDOMString(), ixmlNodeList_free(), and ixmlNodeList_length().

9.120.1.21 RemoveSubscriptionSID()

```
void RemoveSubscriptionSID (
            Upnp_SID sid,
            service_info * service )
```

Parameters

in	*sid*	Subscription ID.
in	*service*	Service object providing the list of subscriptions.

9.121 service_table.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (c) 2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * - Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * - Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * - Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
```

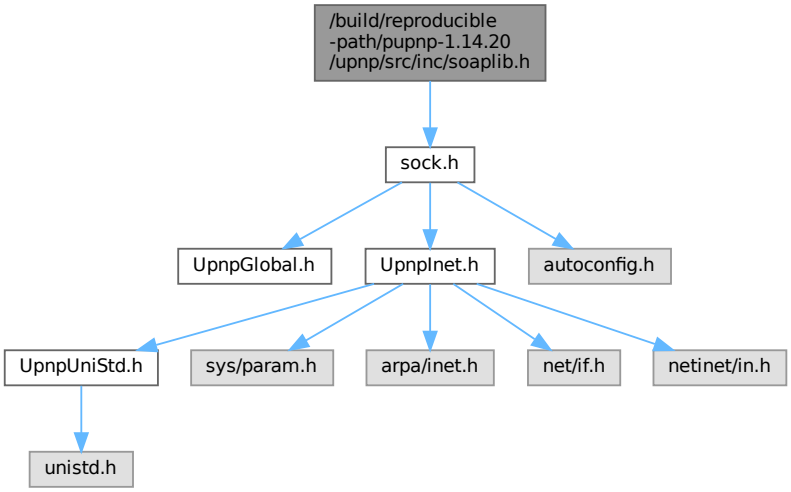
```
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef SERVICE_TABLE_H
00034 #define SERVICE_TABLE_H
00035
00040 #ifdef __cplusplus
00041 extern "C" {
00042 #endif
00043
00044 #include "LinkedList.h"
00045 #include "config.h"
00046 #include "ixml.h"
00047 #include "upnp.h"
00048 #include "upnpdebug.h"
00049 #include "uri.h"
00050
00051 #include <stdio.h>
00052 #include <time.h>
00053
00054 #define SID_SIZE (size_t)41
00055
00056 typedef struct SUBSCRIPTION
00057 {
00058         Upnp_SID sid;
00059         int ToSendEventKey;
00060         time_t expireTime;
00061         int active;
00062         URL_list DeliveryURLs;
00063         /* List of queued events for this subscription. Only one event job
00064            at a time goes into the thread pool. The first element in the
00065            list is a copy of the active job. Others are activated on job
00066            completion. */
00067         LinkedList outgoing;
00068         struct SUBSCRIPTION *next;
00069 } subscription;
00070
00071 typedef struct SERVICE_INFO
00072 {
00073         DOMString serviceType;
00074         DOMString serviceId;
00075         char *SCPDURL;
00076         char *controlURL;
00077         char *eventURL;
00078         DOMString UDN;
00079         int active;
00080         int TotalSubscriptions;
00081         subscription *subscriptionList;
00082         struct SERVICE_INFO *next;
00083 } service_info;
00084
00085 #ifdef INCLUDE_DEVICE_APIS
00086
00087 extern void freeSubscriptionQueuedEvents(subscription *sub);
00088
00089 typedef struct SERVICE_TABLE
00090 {
00091         DOMString URLBase;
00092         service_info *serviceList;
00093         service_info *endServiceList;
00094 } service_table;
00095
00096 /* Functions for Subscriptions */
00097
00103 int copy_subscription(
00105         subscription *in,
00107         subscription *out);
00108
00109 /*
00110  * \brief Remove the subscription represented by the const Upnp_SID sid
00111  * parameter from the service table and update the service table.
00112  */
00113 void RemoveSubscriptionSID(
00115         Upnp_SID sid,
00117         service_info *service);
00118
00125 subscription *GetSubscriptionSID(
00127         const Upnp_SID sid,
00129         service_info *service);
```

```
00130
00136 subscription *GetFirstSubscription(
00138         service_info *service);
00139
00145 subscription *GetNextSubscription(
00147         service_info *service,
00149         subscription *current);
00150
00154 void freeSubscription(
00156         subscription *sub);
00157
00162 void freeSubscriptionList(
00164         subscription *head);
00165
00172 service_info *FindServiceId(
00174         service_table *table,
00177         const char *serviceId,
00180         const char *UDN);
00181
00189 service_info *FindServiceEventURLPath(
00191         service_table *table,
00193         const char *eventURLPath);
00194
00202 service_info *FindServiceControlURLPath(
00204         service_table *table,
00206         const char *controlURLPath);
00207
00212         #ifdef DEBUG
00213 void printService(
00215         service_info *service,
00217         Upnp_LogLevel level,
00219         Dbg_Module module);
00220         #else
00221                 #define printService(service, level, module) \
00222                         do { \
00223                         } while (0)
00224         #endif
00225
00230         #ifdef DEBUG
00231 void printServiceList(
00233         service_info *service,
00235         Upnp_LogLevel level,
00237         Dbg_Module module);
00238         #else
00239                 #define printServiceList(service, level, module) \
00240                         do { \
00241                         } while (0)
00242         #endif
00243
00249         #ifdef DEBUG
00250 void printServiceTable(
00252         service_table *table,
00254         Upnp_LogLevel level,
00256         Dbg_Module module);
00257         #else
00258                 #define printServiceTable(table, level, module) \
00259                         do { \
00260                         } while (0)
00261         #endif
00262
00267 void freeService(
00269         service_info *in);
00270
00275 void freeServiceList(
00277         service_info *head);
00278
00283 void freeServiceTable(
00285         service_table *table);
00286
00295 int removeServiceTable(
00297         IXML_Node *node,
00299         service_table *in);
00300
00304 int addServiceTable(
00306         IXML_Node *node,
00308         service_table *in,
00311         const char *DefaultURLBase);
00312
00318 int getServiceTable(
00320         IXML_Node *node,
00323         service_table *out,
00325         const char *DefaultURLBase);
00326
00327 /*      Misc helper functions   */
00328
00336 DOMString getElementValue(
00338         IXML_Node *node);
```

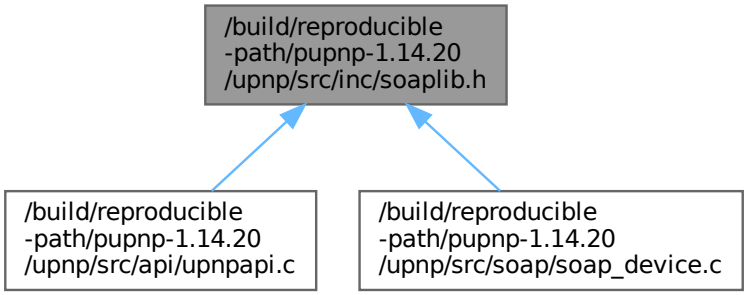
```
00339
00348 int getSubElement(
00350         const char *element_name,
00352         IXML_Node *node,
00354         IXML_Node **out);
00355
00356 #endif /* INCLUDE_DEVICE_APIS */
00357
00358 #ifdef __cplusplus
00359 }
00360 #endif
00361
00362 #endif /* SERVICE_TABLE */
```

9.122 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/soaplib.h File Reference

`#include "sock.h"`
Include dependency graph for soaplib.h:



This graph shows which files directly or indirectly include this file:



Functions

- void soap_device_callback (http_parser_t *parser, http_message_t *request, SOCKINFO *info)

  This is a callback called by minisever after receiving the request from the control point. This function will start processing the request. It calls handle_invoke_action to handle the SOAP action.
- int **SoapSendAction** (char *action_url, char *service_type, IXML_Document *action_node, IXML_Document **response_node)
- int **SoapSendActionEx** (char *ActionURL, char *ServiceType, IXML_Document *Header, IXML_Document *ActNode, IXML_Document **RespNode)
- int **SoapGetServiceVarStatus** (char *ActionURL, DOMString VarName, DOMString *StVar)

Variables

- const char * **ContentTypeHeader**

9.122.1 Function Documentation

9.122.1.1 soap_device_callback()

```
void soap_device_callback (
             http_parser_t * parser,
             http_message_t * request,
             SOCKINFO * info )
```

This is a callback called by minisever after receiving the request from the control point. This function will start processing the request. It calls handle_invoke_action to handle the SOAP action.

This is a callback called by minisever after receiving the request from the control point. This function will start processing the request. It calls handle_invoke_action to handle the SOAP action.

Parameters

in	*parser*	Parsed request received by the device.
in	*request*	HTTP request.
in,out	*info*	Socket info.

References memptr::buf, check_soap_request(), check_soapaction_hdr(), http_message_t::entity, SOCKINFO::foreign_sockaddr, get_dev_service(), handle_invoke_action(), handle_query_variable(), has_xml_content_type(), ixmlDocument_free(), ixmlParseBufferEx(), UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by UpnpInitPreamble().

9.123 soaplib.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
```

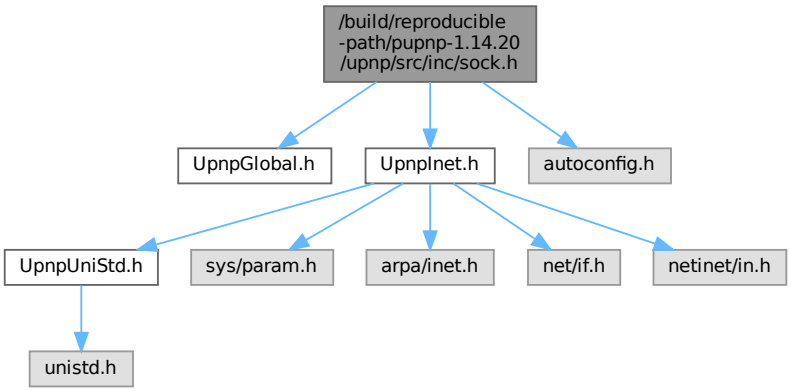
```
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef SOAPLIB_H
00033 #define SOAPLIB_H
00034
00039 /* SOAP module API to be called in Upnp-Dk API */
00040
00041 #include "sock.h"
00042
00048 void soap_device_callback(
00050         http_parser_t *parser,
00052         http_message_t *request,
00054         SOCKINFO *info);
00055
00056 /*****************************************************************************
00057  * Function: SoapSendAction
00058  *
00059  * Parameters:
00060  *      IN char* action_url: device contrl URL
00061  *      IN char *service_type: device service type
00062  *      IN IXML_Document *action_node: SOAP action node
00063  *      OUT IXML_Document **response_node: SOAP response node
00064  *
00065  * Description: This function is called by UPnP API to send the SOAP
00066  *      action request and waits till it gets the response from the device
00067  *      pass the response to the API layer
00068  *
00069  * Return: int
00070  *      returns UPNP_E_SUCCESS if successful else returns appropriate error
00071  * Note:
00072  ****************************************************************************/
00073 int SoapSendAction(char *action_url,
00074         char *service_type,
00075         IXML_Document *action_node,
00076         IXML_Document **response_node);
00077
00078 /*****************************************************************************
00079  * Function: SoapSendActionEx
00080  *
00081  * Parameters:
00082  *      IN char* action_url: device contrl URL
00083  *      IN char *service_type: device service type
00084  *      IN IXML_Document *Header: Soap header
00085  *      IN IXML_Document *action_node: SOAP action node (SOAP body)
00086  *      OUT IXML_Document **response_node: SOAP response node
00087  *
00088  * Description: This function is called by UPnP API to send the SOAP
00089  *      action request and waits till it gets the response from the device
00090  *      pass the response to the API layer. This action is similar to the
00091  *      the SoapSendAction with only difference that it allows users to
00092  *      pass the SOAP header along the SOAP body ( soap action request)
00093  *
00094  * Return: int
00095  *      returns UPNP_E_SUCCESS if successful else returns appropriate error
00096  * Note:
00097  ****************************************************************************/
00098 int SoapSendActionEx(char *ActionURL,
00099         char *ServiceType,
00100         IXML_Document *Header,
00101         IXML_Document *ActNode,
00102         IXML_Document **RespNode);
00103
00104 /*****************************************************************************
00105  * Function: SoapGetServiceVarStatus
```

```
00106  *
00107  * Parameters:
00108  *      IN  char * action_url: Address to send this variable query message.
00109  *      IN  char *var_name: Name of the variable.
00110  *      OUT char **var_value: Output value.
00111  *
00112  * Description: This function creates a status variable query message
00113  *      send it to the specified URL. It also collect the response.
00114  *
00115  * Return: int
00116  *
00117  * Note:
00118  ****************************************************************************/
00119 int SoapGetServiceVarStatus(
00120         char *ActionURL, DOMString VarName, DOMString *StVar);
00121
00122 extern const char *ContentTypeHeader;
00123
00124 #endif /* SOAPLIB_H */
```

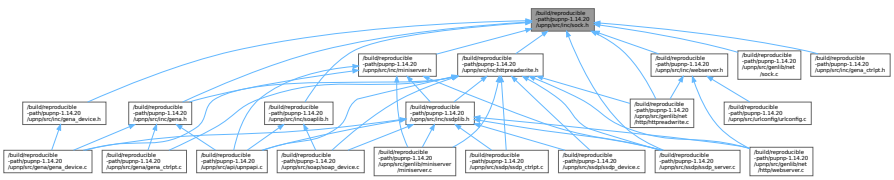
9.124 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sock.h File Reference

```
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include "autoconfig.h"
```

Include dependency graph for sock.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct SOCKINFO

Macros

- #define **SD_RECEIVE** 0x00
- #define **SD_SEND** 0x01
- #define **SD_BOTH** 0x02

Functions

- static UPNP_INLINE int sock_close (SOCKET sock)

  Closes the socket if it is different from -1.
- int sock_init (SOCKINFO *info, SOCKET sockfd)

  Assign the passed in socket descriptor to socket descriptor in the SOCKINFO structure.
- int sock_init_with_ip (SOCKINFO *info, SOCKET sockfd, struct sockaddr *foreign_sockaddr)

  Calls the sock_init function and assigns the passed in IP address and port to the IP address and port in the SOCKINFO structure.
- int sock_destroy (SOCKINFO *info, int ShutdownMethod)

  Associates an SSL object with the socket and begins the client-side SSL/TLS handshake.
- int sock_read (SOCKINFO *info, char *buffer, size_t bufsize, int *timeoutSecs)

  Reads data on socket in sockinfo.
- int sock_write (SOCKINFO *info, const char *buffer, size_t bufsize, int *timeoutSecs)

  Writes data on the socket in sockinfo.
- int sock_make_blocking (SOCKET sock)

  Make socket blocking.
- int sock_make_no_blocking (SOCKET sock)

  Make socket non-blocking.

9.125 sock.h

Go to the documentation of this file.

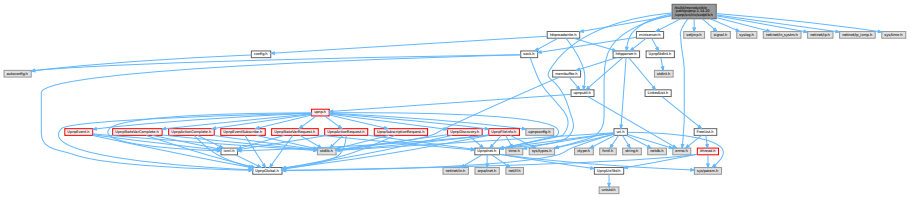
```
00001 #ifndef GENLIB_NET_SOCK_H
00002 #define GENLIB_NET_SOCK_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  * Copyright (c) 2012 France Telecom All rights reserved.
00009  *
00010  * Redistribution and use in source and binary forms, with or without
00011  * modification, are permitted provided that the following conditions are met:
00012  *
00013  * - Redistributions of source code must retain the above copyright notice,
00014  * this list of conditions and the following disclaimer.
00015  * - Redistributions in binary form must reproduce the above copyright notice,
00016  * this list of conditions and the following disclaimer in the documentation
00017  * and/or other materials provided with the distribution.
00018  * - Neither name of Intel Corporation nor the names of its contributors
00019  * may be used to endorse or promote products derived from this software
00020  * without specific prior written permission.
00021  *
00022  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00026  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00027  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00028  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00029  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00030  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00031  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00032  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  ******************************************************************************/
00035
```

```
00044 #include "UpnpGlobal.h" /* for UPNP_INLINE */
00045 #include "UpnpInet.h" /* for SOCKET, netinet/in */
00046 #include "autoconfig.h"
00047 #ifdef UPNP_ENABLE_OPEN_SSL
00048         #include <openssl/ssl.h>
00049 #endif
00050
00051 /* The following are not defined under winsock.h */
00052 #ifndef SD_RECEIVE
00053         #define SD_RECEIVE 0x00
00054         #define SD_SEND 0x01
00055         #define SD_BOTH 0x02
00056 #endif
00057
00059 typedef struct
00060 {
00062         SOCKET socket;
00064         struct sockaddr_storage foreign_sockaddr;
00065 #ifdef UPNP_ENABLE_OPEN_SSL
00066         SSL *ssl;
00067 #endif
00068 } SOCKINFO;
00069
00070 #ifdef __cplusplus
00071 extern "C" {
00072 #endif
00073
00079 static UPNP_INLINE int sock_close(
00081         SOCKET sock)
00082 {
00083         int ret = -1;
00084
00085         if (sock != INVALID_SOCKET)
00086                 ret = UpnpCloseSocket(sock);
00087
00088         return ret;
00089 }
00090
00100 int sock_init(
00102         SOCKINFO *info,
00104         SOCKET sockfd);
00105
00115 int sock_init_with_ip(
00117         SOCKINFO *info,
00119         SOCKET sockfd,
00121         struct sockaddr *foreign_sockaddr);
00122
00131 #ifdef UPNP_ENABLE_OPEN_SSL
00132 int sock_ssl_connect(
00134         SOCKINFO *info);
00135 #endif
00136
00148 int sock_destroy(
00150         SOCKINFO *info,
00152         int ShutdownMethod);
00153
00162 int sock_read(
00164         SOCKINFO *info,
00166         char *buffer,
00168         size_t bufsize,
00170         int *timeoutSecs);
00171
00180 int sock_write(
00182         SOCKINFO *info,
00184         const char *buffer,
00186         size_t bufsize,
00188         int *timeoutSecs);
00189
00195 int sock_make_blocking(
00196         /* [in] socket. */
00197         SOCKET sock);
00198
00204 int sock_make_no_blocking(
00205         /* [in] socket. */
00206         SOCKET sock);
00207
00208 #ifdef __cplusplus
00209 } /* #extern "C" */
00210 #endif
00211
00212 /* @} Sock Network Socket Library */
00213
00214 #endif /* GENLIB_NET_SOCK_H */
```

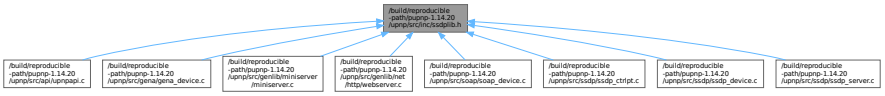
9.126 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h File Reference

```
#include "UpnpInet.h"
#include "httpparser.h"
#include "httpreadwrite.h"
#include "miniserver.h"
#include <errno.h>
#include <setjmp.h>
#include <signal.h>
#include <sys/types.h>
#include <syslog.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <sys/time.h>
```

Include dependency graph for ssdplib.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct SsdpEventStruct
- struct TData
- struct ssdpsearchreply
- struct ssdpsearcharg
- struct ssdpsearchexparg
- struct ssdp_thread_data

Macros

- #define **BUFSIZE** (size_t)2500
- #define **SSDP_IP** "239.255.255.250"
- #define **SSDP_IPV6_LINKLOCAL** "FF02::C"
- #define **SSDP_IPV6_SITELOCAL** "FF05::C"
- #define **SSDP_PORT** 1900
- #define **NUM_TRY** 3
- #define **THREAD_LIMIT** 50
- #define **COMMAND_LEN** 300

Typedefs

- typedef enum SsdpSearchType SType

Enumerations

- enum SsdpSearchType {
  SSDP_SERROR = -1 , **SSDP_ALL** , **SSDP_ROOTDEVICE** , **SSDP_DEVICEUDN** ,
  SSDP_DEVICETYPE , **SSDP_SERVICE** }

Functions

SSDP Server Functions

- int AdvertiseAndReply (int AdFlag, UpnpDevice_Handle Hnd, enum SsdpSearchType SearchType, struct sockaddr *DestAddr, char *DeviceType, char *DeviceUDN, char *ServiceType, int Exp)
  Sends SSDP advertisements, replies and shutdown messages.
- int unique_service_name (char *cmd, SsdpEvent *Evt)
  Fills the fields of the event structure like DeviceType, Device UDN and Service Type.
- enum SsdpSearchType ssdp_request_type1 (char *cmd)
  This function figures out the type of the SSDP search in the in the request.
- int ssdp_request_type (char *cmd, SsdpEvent *Evt)
  Starts filling the SSDP event structure based upon the request received.
- int readFromSSDPSocket (SOCKET socket)
  This function reads the data from the ssdp socket.
- int get_ssdp_sockets (MiniServerSockArray *out)
  Creates the IPv4 and IPv6 ssdp sockets required by the control point and device operation.

SSDP Control Point Functions

- void ssdp_handle_ctrlpt_msg (http_message_t *hmsg, struct sockaddr_storage *dest_addr, int timeout)
  This function handles the ssdp messages from the devices. These messages includes the search replies, advertisement of device coming alive and bye byes.
- int SearchByTarget (int Hnd, int Mx, char *St, void *Cookie)
  Creates and send the search request for a specific URL.

SSDP Device Functions

- void advertiseAndReplyThread (void *data)
  Wrapper function to reply the search request coming from the control point.
- void ssdp_handle_device_request (http_message_t *hmsg, struct sockaddr_storage *dest_addr)
  Handles the search request. It does the sanity checks of the request and then schedules a thread to send a random time reply (random within maximum time given by the control point to reply).
- int DeviceAdvertisement (char *DevType, int RootDev, char *Udn, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)
  Creates the device advertisement request based on the input parameter, and send it to the multicast channel.
- int SendReply (struct sockaddr *DestAddr, char *DevType, int RootDev, char *Udn, char *Location, int Duration, int ByType, int PowerState, int SleepPeriod, int RegistrationState)
  Creates the reply packet based on the input parameter, and send it to the client addesss given in its input parameter DestAddr.
- int DeviceReply (struct sockaddr *DestAddr, char *DevType, int RootDev, char *Udn, char *Location, int Duration, int PowerState, int SleepPeriod, int RegistrationState)
  Creates the reply packet based on the input parameter, and send it to the client address given in its input parameter DestAddr.
- int ServiceAdvertisement (char *Udn, char *ServType, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

Creates the advertisement packet based on the input parameter, and send it to the multicast channel.

- int ServiceReply (struct sockaddr *DestAddr, char *ServType, char *Udn, char *Location, int Duration, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the advertisement packet based on the input parameter, and send it to the multicast channel.
- int ServiceShutdown (char *Udn, char *ServType, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates a HTTP service shutdown request packet and sends it to the multicast channel through RequestHandler.
- int DeviceShutdown (char *DevType, int RootDev, char *Udn, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates a HTTP device shutdown request packet and send it to the multicast channel through RequestHandler.

X_USER_AGENT

can be overwritten by configure CFLAGS argument.

The {\tt X_USER_AGENT} constant specifies the value of the X-User-Agent: HTTP header. The value "redsonic" is needed for the DSM-320. See https://sourceforge.net/forum/message.php?msg_id=3166856 for more information

- #define **X_USER_AGENT** "redsonic"
- #define NO_ERROR_FOUND 0
- #define **E_REQUEST_INVALID** -3
- #define **E_RES_EXPIRED** -4
- #define **E_MEM_ALLOC** -5
- #define **E_HTTP_SYNTEX** -6
- #define **E_SOCKET** -7
- #define **RQST_TIMEOUT** 20
- typedef struct SsdpEventStruct SsdpEvent
- typedef void(* **SsdpFunPtr**) (SsdpEvent *)
- typedef struct TData **ThreadData**
- typedef struct ssdpsearchreply **SsdpSearchReply**
- typedef struct ssdpsearcharg **SsdpSearchArg**
- typedef struct ssdpsearchexparg **SsdpSearchExpArg**
- typedef int(* **ParserFun**) (char *, SsdpEvent *)
- SOCKET **gSsdpReqSocket4**

9.127 ssdplib.h

Go to the documentation of this file.

```
00001 #ifndef SSDPLIB_H
00002 #define SSDPLIB_H
00003
00004 /**************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  * Copyright (C) 2011-2012 France Telecom All rights reserved.
00009  *
00010  * Redistribution and use in source and binary forms, with or without
00011  * modification, are permitted provided that the following conditions are met:
00012  *
00013  * - Redistributions of source code must retain the above copyright notice,
00014  * this list of conditions and the following disclaimer.
00015  * - Redistributions in binary form must reproduce the above copyright notice,
00016  * this list of conditions and the following disclaimer in the documentation
00017  * and/or other materials provided with the distribution.
00018  * - Neither name of Intel Corporation nor the names of its contributors
00019  * may be used to endorse or promote products derived from this software
00020  * without specific prior written permission.
```

```
00021  *
00022  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00026  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00027  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00028  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00029  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00030  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00031  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00032  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  ******************************************************************************/
00035
00044 #include "UpnpInet.h"
00045 #include "httpparser.h"
00046 #include "httpreadwrite.h"
00047 #include "miniserver.h"
00048
00049 #include <errno.h>
00050 #include <setjmp.h>
00051 #include <signal.h>
00052 #include <sys/types.h>
00053
00054 #ifdef _WIN32
00055 #else /* _WIN32 */
00056         #include <syslog.h>
00057         #ifndef __APPLE__
00058                 #include <netinet/in_systm.h>
00059                 #include <netinet/ip.h>
00060                 #include <netinet/ip_icmp.h>
00061         #endif /* __APPLE__ */
00062         #include <sys/time.h>
00063 #endif /* _WIN32 */
00064
00066 typedef enum SsdpSearchType
00067 {
00069         SSDP_SERROR = -1,
00070         SSDP_ALL,
00071         SSDP_ROOTDEVICE,
00072         SSDP_DEVICEUDN,
00073         SSDP_DEVICETYPE,
00074         SSDP_SERVICE
00075 } SType;
00076
00077 #define BUFSIZE (size_t)2500
00078 #define SSDP_IP "239.255.255.250"
00079 #define SSDP_IPV6_LINKLOCAL "FF02::C"
00080 #define SSDP_IPV6_SITELOCAL "FF05::C"
00081 #define SSDP_PORT 1900
00082 #define NUM_TRY 3
00083 #define THREAD_LIMIT 50
00084 #define COMMAND_LEN 300
00085
00087 #ifndef X_USER_AGENT
00094         #define X_USER_AGENT "redsonic"
00095 #endif
00096
00098 #define NO_ERROR_FOUND 0
00099 #define E_REQUEST_INVALID -3
00100 #define E_RES_EXPIRED -4
00101 #define E_MEM_ALLOC -5
00102 #define E_HTTP_SYNTEX -6
00103 #define E_SOCKET -7
00104
00105 #define RQST_TIMEOUT 20
00106
00108 typedef struct SsdpEventStruct
00109 {
00110         enum SsdpSearchType RequestType;
00111         int ErrCode;
00112         int MaxAge;
00113         int Mx;
00114         char UDN[LINE_SIZE];
00115         char DeviceType[LINE_SIZE];
00116         /* NT or ST */
00117         char ServiceType[LINE_SIZE];
00118         char Location[LINE_SIZE];
00119         char HostAddr[LINE_SIZE];
00120         char Os[LINE_SIZE];
00121         char Ext[LINE_SIZE];
00122         char Date[LINE_SIZE];
00123         struct sockaddr *DestAddr;
00124         void *Cookie;
00125 } SsdpEvent;
00126
```

```
00127 typedef void (*SsdpFunPtr)(SsdpEvent *);
00128
00129 typedef struct TData
00130 {
00131         int Mx;
00132         void *Cookie;
00133         char *Data;
00134         struct sockaddr_storage DestAddr;
00135 } ThreadData;
00136
00137 typedef struct ssdpsearchreply
00138 {
00139         int MaxAge;
00140         UpnpDevice_Handle handle;
00141         struct sockaddr_storage dest_addr;
00142         SsdpEvent event;
00143 } SsdpSearchReply;
00144
00145 typedef struct ssdpsearcharg
00146 {
00147         int timeoutEventId;
00148         char *searchTarget;
00149         void *cookie;
00150         enum SsdpSearchType requestType;
00151 } SsdpSearchArg;
00152
00153 typedef struct ssdpsearchexparg
00154 {
00155         int handle;
00156         int timeoutEventId;
00157 } SsdpSearchExpArg;
00158
00159 typedef struct
00160 {
00161         http_parser_t parser;
00162         struct sockaddr_storage dest_addr;
00163 } ssdp_thread_data;
00164
00165 /* globals */
00166
00167 #ifdef INCLUDE_CLIENT_APIS
00168 extern SOCKET gSsdpReqSocket4;
00169         #ifdef UPNP_ENABLE_IPV6
00170 extern SOCKET gSsdpReqSocket6;
00171         #endif /* UPNP_ENABLE_IPV6 */
00172 #endif         /* INCLUDE_CLIENT_APIS */
00173 typedef int (*ParserFun)(char *, SsdpEvent *);
00174
00186 int AdvertiseAndReply(
00187         /* [in] -1 = Send shutdown, 0 = send reply, 1 = Send Advertisement. */
00188         int AdFlag,
00189         /* [in] Device handle. */
00190         UpnpDevice_Handle Hnd,
00191         /* [in] Search type for sending replies. */
00192         enum SsdpSearchType SearchType,
00193         /* [in] Destination address. */
00194         struct sockaddr *DestAddr,
00195         /* [in] Device type. */
00196         char *DeviceType,
00197         /* [in] Device UDN. */
00198         char *DeviceUDN,
00199         /* [in] Service type. */
00200         char *ServiceType,
00201         /* [in] Advertisement age. */
00202         int Exp);
00203
00210 int unique_service_name(
00211         /* [in] Service Name string. */
00212         char *cmd,
00213         /* [out] The SSDP event structure partially filled by all the
00214          * function. */
00215         SsdpEvent *Evt);
00216
00224 enum SsdpSearchType ssdp_request_type1(
00225         /* [in] command came in the ssdp request. */
00226         char *cmd);
00227
00234 int ssdp_request_type(
00235         /* [in] command came in the ssdp request. */
00236         char *cmd,
00237         /* [out] The event structure partially filled by this function. */
00238         SsdpEvent *Evt);
00239
00245 int readFromSSDPSocket(
00246         /* [in] SSDP socket. */
00247         SOCKET socket);
00248
```

```
00255 int get_ssdp_sockets(
00256         /* [out] Array of SSDP sockets. */
00257         MiniServerSockArray *out);
00258
00259 /* @} SSDP Server Functions */
00260
00272 void ssdp_handle_ctrlpt_msg(
00273         /* [in] SSDP message from the device. */
00274         http_message_t *hmsg,
00275         /* [in] Address of the device. */
00276         struct sockaddr_storage *dest_addr,
00277         /* [in] timeout kept by the control point while sending search message.
00278          * Only in search reply. */
00279         int timeout);
00280
00298 int SearchByTarget(
00299         /* [in] The handle of the client performing the search. */
00300         int Hnd,
00301         /* [in] Number of seconds to wait, to collect all the responses. */
00302         int Mx,
00303         /* [in] Search target. */
00304         char *St,
00305         /* [in] Cookie provided by control point application. This cokie will
00306          * be returned to application in the callback. */
00307         void *Cookie);
00308
00309 /* @} SSDP Control Point Functions */
00310
00321 void advertiseAndReplyThread(
00322         /* [in] Structure containing the search request. */
00323         void *data);
00324
00330 #ifdef INCLUDE_DEVICE_APIS
00331 void ssdp_handle_device_request(
00332         /* [in] . */
00333         http_message_t *hmsg,
00334         /* [in] . */
00335         struct sockaddr_storage *dest_addr);
00336 #else  /* INCLUDE_DEVICE_APIS */
00337 static UPNP_INLINE void ssdp_handle_device_request(
00338         /* [in] . */
00339         http_message_t *hmsg,
00340         /* [in] . */
00341         struct sockaddr_storage *dest_addr)
00342 {
00343 }
00344 #endif /* INCLUDE_DEVICE_APIS */
00345
00352 int DeviceAdvertisement(
00353         /* [in] type of the device. */
00354         char *DevType,
00355         /* [in] flag to indicate if the device is root device. */
00356         int RootDev,
00357         /* [in] UDN. */
00358         char *Udn,
00359         /* [in] Location URL. */
00360         char *Location,
00361         /* [in] Service duration in sec. */
00362         int Duration,
00363         /* [in] Device address family. */
00364         int AddressFamily,
00365         /* [in] PowerState as defined by UPnP Low Power. */
00366         int PowerState,
00367         /* [in] SleepPeriod as defined by UPnP Low Power. */
00368         int SleepPeriod,
00369         /* [in] RegistrationState as defined by UPnP Low Power. */
00370         int RegistrationState);
00371
00378 int SendReply(
00379         /* [in] destination IP address. */
00380         struct sockaddr *DestAddr,
00381         /* [in] Device type. */
00382         char *DevType,
00383         /* [in] 1 means root device 0 means embedded device. */
00384         int RootDev,
00385         /* [in] Device UDN. */
00386         char *Udn,
00387         /* [in] Location of Device description document. */
00388         char *Location,
00389         /* [in] Life time of this device. */
00390         int Duration,
00391         /* [in] . */
00392         int ByType,
00393         /* [in] PowerState as defined by UPnP Low Power. */
00394         int PowerState,
00395         /* [in] SleepPeriod as defined by UPnP Low Power. */
00396         int SleepPeriod,
```

```
00397         /* [in] RegistrationState as defined by UPnP Low Power. */
00398         int RegistrationState);
00399
00406 int DeviceReply(
00407         /* [in] destination IP address. */
00408         struct sockaddr *DestAddr,
00409         /* [in] Device type. */
00410         char *DevType,
00411         /* [in] 1 means root device 0 means embedded device. */
00412         int RootDev,
00413         /* [in] Device UDN. */
00414         char *Udn,
00415         /* [in] Location of Device description document. */
00416         char *Location,
00417         /* [in] Life time of this device. */
00418         int Duration,
00419         /* [in] PowerState as defined by UPnP Low Power. */
00420         int PowerState,
00421         /* [in] SleepPeriod as defined by UPnP Low Power. */
00422         int SleepPeriod,
00423         /* [in] RegistrationState as defined by UPnP Low Power. */
00424         int RegistrationState);
00425
00432 int ServiceAdvertisement(
00433         /* [in] Device UDN. */
00434         char *Udn,
00435         /* [in] Service Type. */
00436         char *ServType,
00437         /* [in] Location of Device description document. */
00438         char *Location,
00439         /* [in] Life time of this device. */
00440         int Duration,
00441         /* [in] Device address family. */
00442         int AddressFamily,
00443         /* [in] PowerState as defined by UPnP Low Power. */
00444         int PowerState,
00445         /* [in] SleepPeriod as defined by UPnP Low Power. */
00446         int SleepPeriod,
00447         /* [in] RegistrationState as defined by UPnP Low Power. */
00448         int RegistrationState);
00449
00456 int ServiceReply(
00457         /* [in] . */
00458         struct sockaddr *DestAddr,
00459         /* [in] Service Type. */
00460         char *ServType,
00461         /* [in] Device UDN. */
00462         char *Udn,
00463         /* [in] Location of Device description document. */
00464         char *Location,
00465         /* [in] Life time of this device. */
00466         int Duration,
00467         /* [in] PowerState as defined by UPnP Low Power. */
00468         int PowerState,
00469         /* [in] SleepPeriod as defined by UPnP Low Power. */
00470         int SleepPeriod,
00471         /* [in] RegistrationState as defined by UPnP Low Power. */
00472         int RegistrationState);
00473
00480 int ServiceShutdown(
00481         /* [in] Device UDN. */
00482         char *Udn,
00483         /* [in] Service Type. */
00484         char *ServType,
00485         /* [in] Location of Device description document. */
00486         char *Location,
00487         /* [in] Service duration in sec. */
00488         int Duration,
00489         /* [in] Device address family. */
00490         int AddressFamily,
00491         /* [in] PowerState as defined by UPnP Low Power. */
00492         int PowerState,
00493         /* [in] SleepPeriod as defined by UPnP Low Power. */
00494         int SleepPeriod,
00495         /* [in] RegistrationState as defined by UPnP Low Power. */
00496         int RegistrationState);
00497
00504 int DeviceShutdown(
00505         /* [in] Device Type. */
00506         char *DevType,
00507         /* [in] 1 means root device. */
00508         int RootDev,
00509         /* [in] Device UDN. */
00510         char *Udn,
00511         /* [in] Location URL. */
00512         char *Location,
00513         /* [in] Device duration in sec. */
```

```
00514         int Duration,
00515         /* [in] Device address family. */
00516         int AddressFamily,
00517         /* [in] PowerState as defined by UPnP Low Power. */
00518         int PowerState,
00519         /* [in] SleepPeriod as defined by UPnP Low Power. */
00520         int SleepPeriod,
00521         /* [in] RegistrationState as defined by UPnP Low Power. */
00522         int RegistrationState);
00523
00524 /* @} SSDP Device Functions */
00525
00526 /* @} SSDPlib SSDP Library */
00527
00528 #endif /* SSDPLIB_H */
```

9.128 statcodes.h

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENLIB_NET_HTTP_STATCODES_H
00033 #define GENLIB_NET_HTTP_STATCODES_H
00034
00035 /* HTTP response status codes */
00036
00037 #define HTTP_CONTINUE 100
00038 #define HTTP_SWITCHING_PROCOTOLS 101
00039
00040 #define HTTP_OK 200
00041 #define HTTP_CREATED 201
00042 #define HTTP_ACCEPTED 202
00043 #define HTTP_NON_AUTHORATATIVE 203
00044 #define HTTP_NO_CONTENT 204
00045 #define HTTP_RESET_CONTENT 205
00046 #define HTTP_PARTIAL_CONTENT 206
00047
00048 #define HTTP_MULTIPLE_CHOICES 300
00049 #define HTTP_MOVED_PERMANENTLY 301
00050 #define HTTP_FOUND 302
00051 #define HTTP_SEE_OTHER 303
00052 #define HTTP_NOT_MODIFIED 304
00053 #define HTTP_USE_PROXY 305
00054 #define HTTP_UNUSED_3XX 306
00055 #define HTTP_TEMPORARY_REDIRECT 307
00056
00057 #define HTTP_BAD_REQUEST 400
00058 #define HTTP_UNAUTHORIZED 401
00059 #define HTTP_PAYMENT_REQD 402
00060 #define HTTP_FORBIDDEN 403
00061 #define HTTP_NOT_FOUND 404
00062 #define HTTP_METHOD_NOT_ALLOWED 405
00063 #define HTTP_NOT_ACCEPTABLE 406
00064 #define HTTP_PROXY_AUTH_REQD 407
00065 #define HTTP_REQUEST_TIMEOUT 408
```

```
00066 #define HTTP_CONFLICT 409
00067 #define HTTP_GONE 410
00068 #define HTTP_LENGTH_REQUIRED 411
00069 #define HTTP_PRECONDITION_FAILED 412
00070 #define HTTP_REQ_ENTITY_TOO_LARGE 413
00071 #define HTTP_REQ_URI_TOO_LONG 414
00072 #define HTTP_UNSUPPORTED_MEDIA_TYPE 415
00073 #define HTTP_REQUEST_RANGE_NOT_SATISFIABLE 416
00074 #define HTTP_EXPECTATION_FAILED 417
00075
00076 #define HTTP_INTERNAL_SERVER_ERROR 500
00077 #define HTTP_NOT_IMPLEMENTED 501
00078 #define HTTP_BAD_GATEWAY 502
00079 #define HTTP_SERVICE_UNAVAILABLE 503
00080 #define HTTP_GATEWAY_TIMEOUT 504
00081 #define HTTP_HTTP_VERSION_NOT_SUPPORTED 505
00082 #define HTTP_VARIANT_ALSO_NEGOTIATES 506
00083 #define HTTP_INSUFFICIENT_STORAGE 507
00084 #define HTTP_LOOP_DETECTED 508
00085 #define HTTP_NOT_EXTENDED 510
00086
00087 /* HTTP lib error codes */
00088
00089 #define HTTP_E_OUT_OF_MEMORY -2
00090 #define HTTP_E_BAD_MSG_FORMAT -3
00091 #define HTTP_E_TIMEDOUT -4
00092 #define HTTP_E_FILE_READ -5
00093
00094 #ifdef __cplusplus
00095 extern "C" {
00096 #endif
00097
00098 /*************************************************************************
00099  * Function: http_get_code_text
00100  *
00101  * Parameters:
00102  *      int statusCode ; Status code based on which the status table and
00103  *                                      status message is returned
00104  *
00105  * Description: Return the right status message based on the passed in
00106  *      int statusCode input parameter
00107  *
00108  * Returns:
00109  *       const char* ptr - pointer to the status message string
00110  *************************************************************************/
00111 const char *http_get_code_text(int statusCode);
00112
00113 #ifdef __cplusplus
00114 } /* extern C */
00115 #endif
00116
00117 #endif /* GENLIB_NET_HTTP_STATCODES_H */
```

9.129 strintmap.h

```
00001 /*****************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

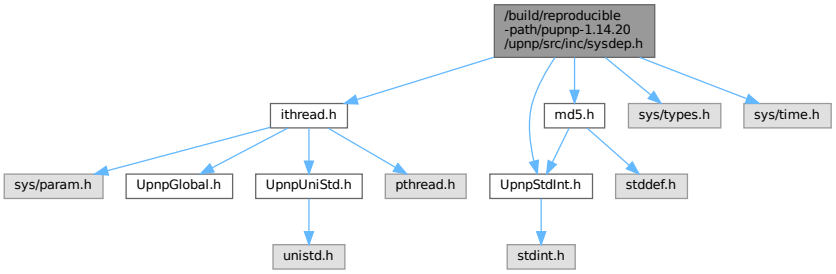
```
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENLIB_UTIL_STRINTMAP_H
00033 #define GENLIB_UTIL_STRINTMAP_H
00034
00035 #include "upnputil.h"
00036 #include <stdlib.h>
00037
00038 /* Util to map from a string to an integer and vice versa */
00039
00040 typedef struct /* str_int_entry */
00041 {
00042         const char *name; /* a value in string form */
00043         int id;           /* same value in integer form */
00044 } str_int_entry;
00045
00046 #ifdef __cplusplus
00047 extern "C" {
00048 #endif
00049
00050 /************************************************************************
00051  *      Function :      map_str_to_int
00052  *
00053  *      Parameters :
00054  *              IN const char* name ;   string containing the name to be matched
00055  *              IN size_t name_len ;    size of the string to be matched
00056  *              IN str_int_entry* table ;       table of entries that need to be
00057  *                                      matched.
00058  *              IN int num_entries ; number of entries in the table that need
00059  *                                      to be searched.
00060  *              IN int case_sensitive ; whether the case should be case
00061  *                                      sensitive or not
00062  *
00063  *      Description : Match the given name with names from the entries in the
00064  *              table. Returns the index of the table when the entry is found.
00065  *
00066  *      Return : int ;
00067  *              index - On Success
00068  *              -1 - On failure
00069  *
00070  *      Note :
00071  ************************************************************************/
00072 int map_str_to_int(const char *name,
00073         size_t name_len,
00074         str_int_entry *table,
00075         int num_entries,
00076         int case_sensitive);
00077
00078 /************************************************************************
00079  *      Function :      map_int_to_str
00080  *
00081  *      Parameters :
00082  *              IN int id ;     ID to be matched
00083  *              IN str_int_entry* table ;       table of entries that need to be
00084  *                                      matched.
00085  *              IN int num_entries ; number of entries in the table that need
00086  *                                      to be searched.
00087  *
00088  *      Description : Returns the index from the table where the id matches
00089  *              the entry from the table.
00090  *
00091  *      Return : int ;
00092  *
00093  *      Note :
00094  ************************************************************************/
00095 int map_int_to_str(int id, str_int_entry *table, int num_entries);
00096
00097 #ifdef __cplusplus
00098 } /* extern C */
00099 #endif
00100
00101 #endif /* GENLIB_UTIL_STRINTMAP_H */
```

9.130 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sysdep.h File Reference

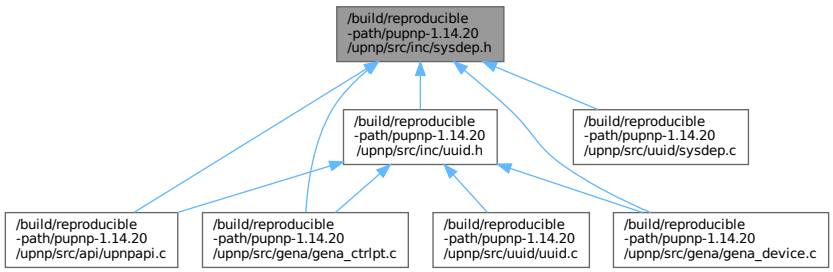
```
#include "ithread.h"
#include "md5.h"
```

```
#include "UpnpStdInt.h"
#include <sys/types.h>
#include <sys/time.h>
```

Include dependency graph for sysdep.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct uuid_node_t

Macros

- #define UUIDS_PER_TICK 1024
- #define **UUIDLock**() ithread_mutex_lock(&gUUIDMutex)
- #define **UUIDUnlock**() ithread_mutex_unlock(&gUUIDMutex)

Typedefs

- typedef uint64_t **uuid_time_t**

Functions

- void get_ieee_node_identifier (uuid_node_t *node)

  System dependent call to get IEEE node ID.
- void get_system_time (uuid_time_t *uuid_time)

  System dependent call to get the current system time.
- void **get_random_info** (unsigned char seed[16])

Variables

- ithread_mutex_t gUUIDMutex

9.130.1 Macro Definition Documentation

9.130.1.1 UUIDS_PER_TICK

```
#define UUIDS_PER_TICK 1024
```

set the following to the number of 100ns ticks of the actual resolution of your system's clock

9.130.2 Function Documentation

9.130.2.1 get_ieee_node_identifier()

```
void get_ieee_node_identifier (
            uuid_node_t * node )
```

System dependent call to get IEEE node ID.

This sample implementation generates a random node ID.

Referenced by uuid_create().

9.130.2.2 get_system_time()

```
void get_system_time (
            uuid_time_t * uuid_time )
```

System dependent call to get the current system time.

Returned as 100ns ticks since Oct 15, 1582, but resolution may be less than 100ns.

Referenced by get_current_time(), and true_random().

9.131 sysdep.h

Go to the documentation of this file.

```
00001 #ifndef SYSDEP_H
00002 #define SYSDEP_H
00003
00004 /*
00005  * Copyright (c) 1990- 1993, 1996 Open Software Foundation, Inc.
00006  * Copyright (c) 1989 by Hewlett-Packard Company, Palo Alto, Ca. &
00007  * Digital Equipment Corporation, Maynard, Mass.
00008  * Copyright (c) 1998 Microsoft.
00009  * To anyone who acknowledges that this file is provided "AS IS"
00010  * without any express or implied warranty: permission to use, copy,
00011  * modify, and distribute this file for any purpose is hereby
00012  * granted without fee, provided that the above copyright notices and
00013  * this notice appears in all source code copies, and that none of
00014  * the names of Open Software Foundation, Inc., Hewlett-Packard
00015  * Company, or Digital Equipment Corporation be used in advertising
00016  * or publicity pertaining to distribution of the software without
00017  * specific, written prior permission.  Neither Open Software
00018  * Foundation, Inc., Hewlett-Packard Company, Microsoft, nor Digital Equipment
00019  * Corporation makes any representations about the suitability of
00020  * this software for any purpose.
00021  */
00022
00027 #include "ithread.h"
00028
00029 /* change to point to where MD5 .h's live */
00030 /* get MD5 sample implementation from RFC 1321 */
00031 #include "md5.h"
00032
00033 #include "UpnpStdInt.h"
00034
00035 #include <sys/types.h>
00036
00037 #ifdef _WIN32
00038         /* Do not #include <sys/time.h> */
00039 #else
00040         #include <sys/time.h>
00041 #endif
00042
00045 #define UUIDS_PER_TICK 1024
00046
00048 extern ithread_mutex_t gUUIDMutex;
00049
00050 #define UUIDLock() ithread_mutex_lock(&gUUIDMutex)
00051 #define UUIDUnlock() ithread_mutex_unlock(&gUUIDMutex)
00052
00053 typedef uint64_t uuid_time_t;
00054
00055 typedef struct
00056 {
00057         char nodeID[6];
00058 } uuid_node_t;
00059
00060 void get_ieee_node_identifier(uuid_node_t *node);
00061 void get_system_time(uuid_time_t *uuid_time);
00062 void get_random_info(unsigned char seed[16]);
00063
00064 #endif /* SYSDEP_H */
```

9.132 unixutil.h

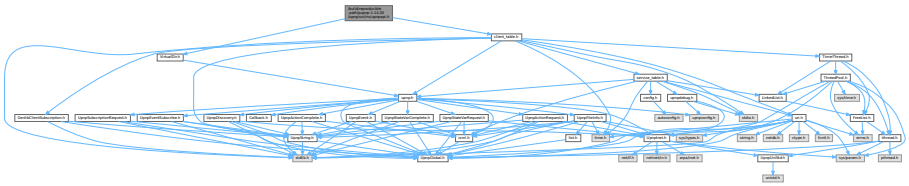
```
00001 #ifndef GENLIB_NET_UNIXUTIL_H
00002 #define GENLIB_NET_UNIXUTIL_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  *
00009  * Redistribution and use in source and binary forms, with or without
00010  * modification, are permitted provided that the following conditions are met:
00011  *
00012  * - Redistributions of source code must retain the above copyright notice,
00013  * this list of conditions and the following disclaimer.
00014  * - Redistributions in binary form must reproduce the above copyright notice,
00015  * this list of conditions and the following disclaimer in the documentation
00016  * and/or other materials provided with the distribution.
00017  * - Neither name of Intel Corporation nor the names of its contributors
00018  * may be used to endorse or promote products derived from this software
```

```
00019  * without specific prior written permission.
00020  *
00021  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00022  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00023  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00024  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00025  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00026  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00027  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00028  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00029  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00030  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00031  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00032  *
00033  ******************************************************************************/
00034
00035 /* Unix-specific network utilities */
00036
00037 #include "UpnpInet.h"
00038
00039 #ifdef _WIN32
00040         #include <UpnpStdInt.h>
00041         #include <errno.h>
00042
00043 typedef int socklen_t;
00044 typedef uint16_t in_port_t;
00045         #ifndef EAFNOSUPPORT /* VS2010 has this defined */
00046                 #define EAFNOSUPPORT 97
00047         #endif
00048 #else
00049         #include <sys/time.h>
00050         #include <sys/wait.h>
00051 #endif
00052
00053 #endif /* GENLIB_NET_UNIXUTIL_H */
```

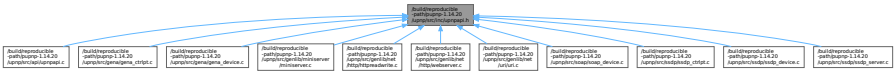
9.133 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnpapi.h File Reference

```
#include "VirtualDir.h"
#include "client_table.h"
```

Include dependency graph for upnpapi.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct Handle_Info
- struct UpnpNonblockParam

Macros

- #define **MAX_INTERFACES** 256
- #define **DEV_LIMIT** 200
- #define **DEFAULT_MX** 5
- #define **DEFAULT_MAXAGE** 1800
- #define **DEFAULT_SOAP_CONTENT_LENGTH** 16000
- #define **MAX_SOAP_CONTENT_LENGTH** (size_t)32000
- #define **NUM_HANDLE** 200
- #define **UPNP_TIMEOUT** 30
- #define **HandleLock**() HandleWriteLock()
- #define HandleWriteLock()
- #define HandleReadLock()
- #define HandleUnlock()
- #define **E_HTTP_SYNTAX** -6

Enumerations

- enum **Upnp_Handle_Type** { **HND_INVALID** = -1 , **HND_CLIENT** , **HND_DEVICE** }
- enum **UpnpFunName** {
  SUBSCRIBE , **UNSUBSCRIBE** , **DK_NOTIFY** , **QUERY** ,
  ACTION , **STATUS** , **DEVDESCRIPTION** , **SERVDESCRIPTION** ,
  MINI , **RENEW** }
- enum **WebServerState** { **WEB_SERVER_DISABLED** , **WEB_SERVER_ENABLED** }

Functions

- Upnp_Handle_Type GetHandleInfo (int Hnd, struct Handle_Info **HndInfo)

  Get handle information.
- Upnp_Handle_Type GetClientHandleInfo (int *client_handle_out, struct Handle_Info **HndInfo)

  Get client handle info.
- Upnp_Handle_Type GetDeviceHandleInfo (UpnpDevice_Handle start, int AddressFamily, int *device_handle_out, struct Handle_Info **HndInfo)

  Retrieves the device handle and information of the first device of the address family specified. The search begins at the 'start' index, which should be 0 for the first call, then the last successful value returned. This allows listing all entries for the address family.
- Upnp_Handle_Type GetDeviceHandleInfoForPath (const char *path, int AddressFamily, int *device_handle_out, struct Handle_Info **HndInfo, service_info **serv_info)

  Retrieves the device handle and information of the first device of the address family specified, with a service having a controlURL or eventSubURL matching the path.
- int UpnpGetIfInfo (const char *IfName)

  Retrieve interface information and keep it in global variables. If NULL, we'll find the first suitable interface for operation.
- void UpnpThreadDistribution (struct UpnpNonblockParam *Param)

  Schedule async functions in threadpool.
- void AutoAdvertise (void *input)

  This function is a timer thread scheduled by UpnpSendAdvertisement to the send advetisement again.
- int PrintHandleInfo (UpnpClient_Handle Hnd)

  Print handle info.

Variables

- size_t g_maxContentLength
- int g_UpnpSdkEQMaxLen
- int g_UpnpSdkEQMaxAge
- ithread_rwlock_t GlobalHndRWLock
- char gIF_NAME [LINE_SIZE]
- char gIF_IPV4 [INET_ADDRSTRLEN]
- char gIF_IPV4_NETMASK [INET_ADDRSTRLEN]
- char gIF_IPV6 [INET6_ADDRSTRLEN]
- unsigned gIF_IPV6_PREFIX_LENGTH
- char gIF_IPV6_ULA_GUA [INET6_ADDRSTRLEN]
- unsigned gIF_IPV6_ULA_GUA_PREFIX_LENGTH
- unsigned gIF_INDEX
- unsigned short LOCAL_PORT_V4
- unsigned short LOCAL_PORT_V6
- unsigned short LOCAL_PORT_V6_ULA_GUA
- Upnp_SID gUpnpSdkNLSuuid
- TimerThread gTimerThread
- ThreadPool gRecvThreadPool
- ThreadPool gSendThreadPool
- ThreadPool gMiniServerThreadPool
- virtualDirList * pVirtualDirList
- struct VirtualDirCallbacks virtualDirCallback
- WebServerState bWebServerState
- WebCallback_HostValidate gWebCallback_HostValidate
- void * gWebCallback_HostValidateCookie
- int gAllowLiteralHostRedirection

9.133.1 Macro Definition Documentation

9.133.1.1 HandleReadLock

```
#define HandleReadLock( )
```

Value:

```
        UpnpPrintf( \
                UPNP_INFO, API, __FILE__, __LINE__, "Trying a read lock\n"); \
        ithread_rwlock_rdlock(&GlobalHndRWLock); \
        UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Read lock acquired\n");
```

9.133.1.2 HandleUnlock

```
#define HandleUnlock( )
```

Value:

```
        UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Trying Unlock\n"); \
        ithread_rwlock_unlock(&GlobalHndRWLock); \
        UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Unlocked rwlock\n");
```

9.133.1.3 HandleWriteLock

```
#define HandleWriteLock( )
```

Value:

```
        UpnpPrintf( \
                UPNP_INFO, API, __FILE__, __LINE__, "Trying a write lock\n"); \
        ithread_rwlock_wrlock(&GlobalHndRWLock); \
        UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Write lock acquired\n");
```

9.133.2 Variable Documentation

9.133.2.1 gUpnpSdkNLSuuid

```
Upnp_SID gUpnpSdkNLSuuid  [extern]
```

NLS uuid.

Referenced by CreateServicePacket(), and UpnpInitPreamble().

9.134 upnpapi.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (C) 2011-2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * * Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * * Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * * Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef UPNPAPI_H
00034 #define UPNPAPI_H
00035
00040 #include "VirtualDir.h" /* for struct VirtualDirCallbacks */
00041 #include "client_table.h"
00042
00043 #define MAX_INTERFACES 256
00044
00045 #define DEV_LIMIT 200
00046
00047 #define DEFAULT_MX 5
00048
00049 #define DEFAULT_MAXAGE 1800
00050
00051 #define DEFAULT_SOAP_CONTENT_LENGTH 16000
```

```
00052 #define MAX_SOAP_CONTENT_LENGTH (size_t)32000
00053
00054 #define NUM_HANDLE 200
00055
00056 extern size_t g_maxContentLength;
00057 extern int g_UpnpSdkEQMaxLen;
00058 extern int g_UpnpSdkEQMaxAge;
00059
00060 /* 30-second timeout */
00061 #define UPNP_TIMEOUT 30
00062
00063 typedef enum
00064 {
00065         HND_INVALID = -1,
00066         HND_CLIENT,
00067         HND_DEVICE
00068 } Upnp_Handle_Type;
00069
00070 /* Data to be stored in handle table for */
00071 struct Handle_Info
00072 {
00074         Upnp_Handle_Type HType;
00076         Upnp_FunPtr Callback;
00078         char *Cookie;
00080         int aliasInstalled;
00081
00082         /* Device Only */
00083 #ifdef INCLUDE_DEVICE_APIS
00085         char DescURL[LINE_SIZE];
00088         char LowerDescURL[LINE_SIZE];
00090         char DescXML[LINE_SIZE];
00091         /* Advertisement timeout */
00092         int MaxAge;
00093         /* Power State as defined by UPnP Low Power. */
00094         int PowerState;
00095         /* Sleep Period as defined by UPnP Low Power. */
00096         int SleepPeriod;
00097         /* Registration State as defined by UPnP Low Power. */
00098         int RegistrationState;
00100         IXML_Document *DescDocument;
00102         IXML_NodeList *DeviceList;
00104         IXML_NodeList *ServiceList;
00106         service_table ServiceTable;
00108         int MaxSubscriptions;
00110         int MaxSubscriptionTimeOut;
00112         int DeviceAf;
00113 #endif
00114
00115         /* Client only */
00116 #ifdef INCLUDE_CLIENT_APIS
00118         GenlibClientSubscription *ClientSubList;
00120         LinkedList SsdpSearchList;
00121 #endif
00122 };
00123
00124 extern ithread_rwlock_t GlobalHndRWLock;
00125
00131 Upnp_Handle_Type GetHandleInfo(
00133         int Hnd,
00135         struct Handle_Info **HndInfo);
00136
00137 #define HandleLock() HandleWriteLock()
00138
00139 #define HandleWriteLock() \
00140         UpnpPrintf( \
00141                 UPNP_INFO, API, __FILE__, __LINE__, "Trying a write lock\n"); \
00142         ithread_rwlock_wrlock(&GlobalHndRWLock); \
00143         UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Write lock acquired\n");
00144
00145 #define HandleReadLock() \
00146         UpnpPrintf( \
00147                 UPNP_INFO, API, __FILE__, __LINE__, "Trying a read lock\n"); \
00148         ithread_rwlock_rdlock(&GlobalHndRWLock); \
00149         UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Read lock acquired\n");
00150
00151 #define HandleUnlock() \
00152         UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Trying Unlock\n"); \
00153         ithread_rwlock_unlock(&GlobalHndRWLock); \
00154         UpnpPrintf(UPNP_INFO, API, __FILE__, __LINE__, "Unlocked rwlock\n");
00155
00163 Upnp_Handle_Type GetClientHandleInfo(
00165         int *client_handle_out,
00167         struct Handle_Info **HndInfo);
00176 Upnp_Handle_Type GetDeviceHandleInfo(
00178         UpnpDevice_Handle start,
00180         int AddressFamily,
00182         int *device_handle_out,
```

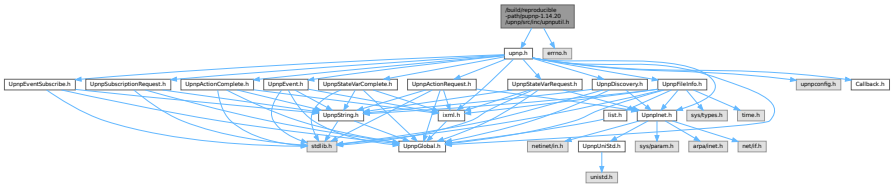
```
00184         struct Handle_Info **HndInfo);
00185
00193 Upnp_Handle_Type GetDeviceHandleInfoForPath(
00195         const char *path,
00197         int AddressFamily,
00199         int *device_handle_out,
00201         struct Handle_Info **HndInfo,
00203         service_info **serv_info);
00204
00205 extern char gIF_NAME[LINE_SIZE];
00206 extern char gIF_IPV4[INET_ADDRSTRLEN];
00207 extern char gIF_IPV4_NETMASK[INET_ADDRSTRLEN];
00208 extern char gIF_IPV6[INET6_ADDRSTRLEN];
00209 extern unsigned gIF_IPV6_PREFIX_LENGTH;
00210
00211 extern char gIF_IPV6_ULA_GUA[INET6_ADDRSTRLEN];
00212 extern unsigned gIF_IPV6_ULA_GUA_PREFIX_LENGTH;
00213
00214 extern unsigned gIF_INDEX;
00215
00216 extern unsigned short LOCAL_PORT_V4;
00217 extern unsigned short LOCAL_PORT_V6;
00218 extern unsigned short LOCAL_PORT_V6_ULA_GUA;
00219
00221 extern Upnp_SID gUpnpSdkNLSuuid;
00222
00223 extern TimerThread gTimerThread;
00224 extern ThreadPool gRecvThreadPool;
00225 extern ThreadPool gSendThreadPool;
00226 extern ThreadPool gMiniServerThreadPool;
00227
00228 typedef enum
00229 {
00230         SUBSCRIBE,
00231         UNSUBSCRIBE,
00232         DK_NOTIFY,
00233         QUERY,
00234         ACTION,
00235         STATUS,
00236         DEVDESCRIPTION,
00237         SERVDESCRIPTION,
00238         MINI,
00239         RENEW
00240 } UpnpFunName;
00241
00242 struct UpnpNonblockParam
00243 {
00244         UpnpFunName FunName;
00245         int Handle;
00246         int TimeOut;
00247         char VarName[NAME_SIZE];
00248         char NewVal[NAME_SIZE];
00249         char DevType[NAME_SIZE];
00250         char DevId[NAME_SIZE];
00251         char ServiceType[NAME_SIZE];
00252         char ServiceVer[NAME_SIZE];
00253         char Url[NAME_SIZE];
00254         Upnp_SID SubsId;
00255         char *Cookie;
00256         Upnp_FunPtr Fun;
00257         IXML_Document *Header;
00258         IXML_Document *Act;
00259         struct DevDesc *Devdesc;
00260 };
00261
00262 extern virtualDirList *pVirtualDirList;
00263 extern struct VirtualDirCallbacks virtualDirCallback;
00264
00265 typedef enum
00266 {
00267         WEB_SERVER_DISABLED,
00268         WEB_SERVER_ENABLED
00269 } WebServerState;
00270
00271 #define E_HTTP_SYNTAX -6
00272
00292 int UpnpGetIfInfo(
00294         const char *IfName);
00295
00296 void UpnpThreadDistribution(struct UpnpNonblockParam *Param);
00297
00302 void AutoAdvertise(
00304         void *input);
00305
00311 int PrintHandleInfo(
00313         UpnpClient_Handle Hnd);
00314
```

```
00316 extern WebServerState bWebServerState;
00317
00319 extern WebCallback_HostValidate gWebCallback_HostValidate;
00320
00322 extern void *gWebCallback_HostValidateCookie;
00323
00325 extern int gAllowLiteralHostRedirection;
00326
00327 #endif /* UPNPAPI_H */
```

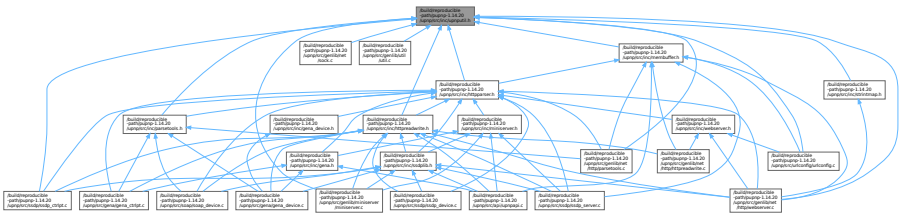
9.135 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnputil.h File Reference

```
#include "upnp.h"
#include <errno.h>
```

Include dependency graph for upnputil.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **GEMD_OUT_OF_MEMORY** -1
- #define **EVENT_TIMEDOUT** -2
- #define **EVENT_TERMINATE** -3
- #define **ERROR_BUFFER_LEN** (size_t)256
- #define **max**(a, b) (((a) > (b)) ? (a) : (b))
- #define **min**(a, b) (((a) < (b)) ? (a) : (b))

Functions

- void linecopy (char dest[LINE_SIZE], const char *src)

  Copy no of bytes spcified by the LINE_SIZE constant, from the source buffer. Null terminate the destination buffer.
- void namecopy (char dest[NAME_SIZE], const char *src)

  Copy no of bytes spcified by the NAME_SIZE constant, from the source buffer. Null terminate the destination buffer.
- void linecopylen (char dest[LINE_SIZE], const char *src, size_t srclen)

  Determine if the srclen passed in paramter is less than the permitted LINE_SIZE. If it is use the passed parameter, if not use the permitted LINE_SIZE as the length parameter.

9.135.1 Function Documentation

9.135.1.1 linecopy()

```
void linecopy (
            char dest[LINE_SIZE],
            const char * src )
```

Copy no of bytes spcified by the LINE_SIZE constant, from the source buffer. Null terminate the destination buffer.

Parameters

out	*dest*	output buffer.
in	*src*	input buffer.

9.135.1.2 linecopylen()

```
void linecopylen (
            char dest[LINE_SIZE],
            const char * src,
            size_t srclen )
```

Determine if the srclen passed in paramter is less than the permitted LINE_SIZE. If it is use the passed parameter, if not use the permitted LINE_SIZE as the length parameter.

Copy no of bytes spcified by the LINE_SIZE constant, from the source buffer. Null terminate the destination buffer.

Parameters

out	*dest*	output buffer.
in	*src*	input buffer.
in	*srclen*	bytes to be copied.

9.135.1.3 namecopy()

```
void namecopy (
            char dest[NAME_SIZE],
            const char * src )
```

Copy no of bytes spcified by the NAME_SIZE constant, from the source buffer. Null terminate the destination buffer.

Parameters

out	*dest*	output buffer.
in	*src*	input buffer.

Referenced by check_soapaction_hdr(), and get_dev_service().

9.136 upnputil.h

Go to the documentation of this file.

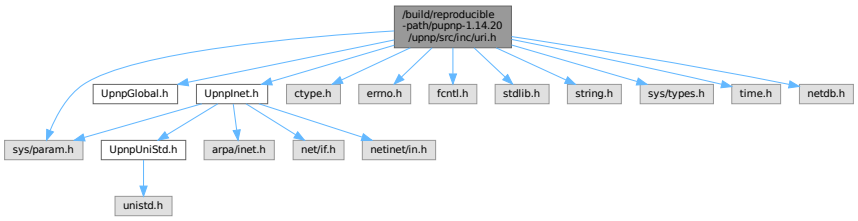
```
00001 #ifndef UTIL_H
00002 #define UTIL_H
00003
00004 /*******************************************************************************
00005  *
00006  * Copyright (c) 2000-2003 Intel Corporation
00007  * All rights reserved.
00008  * Copyright (c) 2012 France Telecom All rights reserved.
00009  *
00010  * Redistribution and use in source and binary forms, with or without
00011  * modification, are permitted provided that the following conditions are met:
00012  *
00013  * - Redistributions of source code must retain the above copyright notice,
00014  * this list of conditions and the following disclaimer.
00015  * - Redistributions in binary form must reproduce the above copyright notice,
00016  * this list of conditions and the following disclaimer in the documentation
00017  * and/or other materials provided with the distribution.
00018  * - Neither name of Intel Corporation nor the names of its contributors
00019  * may be used to endorse or promote products derived from this software
00020  * without specific prior written permission.
00021  *
00022  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00026  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00027  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00028  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00029  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00030  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00031  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00032  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  ******************************************************************************/
00035
00040 #include "upnp.h"
00041 #include <errno.h>
00042
00043 #define GEMD_OUT_OF_MEMORY -1
00044 #define EVENT_TIMEDOUT -2
00045 #define EVENT_TERMINATE -3
00046
00047 #ifdef __cplusplus
00048 extern "C" {
00049 #endif
00050
00055 void linecopy(
00057         char dest[LINE_SIZE],
00059         const char *src);
00060
00065 void namecopy(
00067         char dest[NAME_SIZE],
00069         const char *src);
00070
00079 void linecopylen(
00081         char dest[LINE_SIZE],
00083         const char *src,
00085         size_t srclen);
00086
00087 #ifdef __cplusplus
00088 }
00089 #endif
00090
00091 /* Size of the errorBuffer variable, passed to the strerror_r() function */
00092 #define ERROR_BUFFER_LEN (size_t)256
00093
00094 /* C specific */
00095 /* VC needs these in C++ mode too (do other compilers?) */
00096 #if !defined(__cplusplus) || defined(UPNP_USE_MSVCPP)
00097         #ifdef _WIN32
00098                 #ifndef S_ISREG
00099                         #define S_ISREG(m) (((m)&S_IFMT) == S_IFREG)
00100                 #endif
00101                 #ifndef S_ISDIR
00102                         #define S_ISDIR(m) (((m)&S_IFMT) == S_IFDIR)
00103                 #endif
00104                 #ifndef EADDRINUSE /* VS2010 has this defined */
00105                         #define EADDRINUSE WSAEADDRINUSE
00106                 #endif
00107                 #define sleep(a) Sleep((a)*1000)
00108                 #define usleep(a) Sleep((a) / 1000)
00109                 #define strerror_r(a, b, c) (strerror_s((b), (c), (a)))
```

```
00110         #else
00111                 #define max(a, b) (((a) > (b)) ? (a) : (b))
00112                 #define min(a, b) (((a) < (b)) ? (a) : (b))
00113         #endif /* _WIN32 */
00114 #endif         /* !defined(__cplusplus) || defined(UPNP_USE_MSVCPP) */
00115
00116 #endif /* UTIL_H */
```

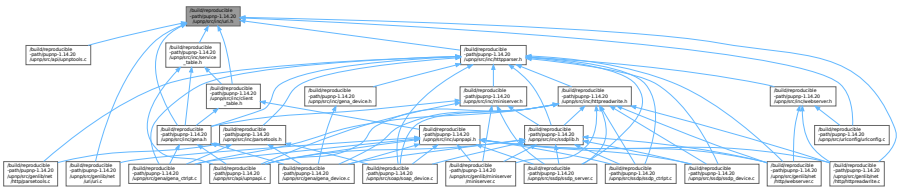
9.137 /build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h File Reference

```
#include <sys/param.h>
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include <ctype.h>
#include <errno.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <netdb.h>
```

Include dependency graph for uri.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct TOKEN

  Buffer used in parsinghttp messages, urls, etc. generally this simply holds a pointer into a larger array.
- struct HOSTPORT

  Represents a host port: e.g. "127.127.0.1:80" text is a token pointing to the full string representation.
- struct URI

  Represents a URI used in parse_uri and elsewhere.
- struct URL_LIST

  *Represents a list of URLs as in the "callback" header of SUBSCRIBE message in GENA. "char * " URLs holds dynamic memory.*

Macros

- #define **MARK** "-_.!~*'()"
- #define RESERVED ";/?:@&=+$,{}"
- #define **HTTP_SUCCESS** 1

Typedefs

- typedef struct TOKEN token

  Buffer used in parsinghttp messages, urls, etc. generally this simply holds a pointer into a larger array.
- typedef struct HOSTPORT hostport_type

  Represents a host port: e.g. "127.127.0.1:80" text is a token pointing to the full string representation.
- typedef struct URI uri_type

  Represents a URI used in parse_uri and elsewhere.
- typedef struct URL_LIST URL_list

  Represents a list of URLs as in the "callback" header of SUBSCRIBE message in GENA. "char " URLs holds dynamic memory.*

Enumerations

- enum **hostType** { **HOSTNAME** , **IPv4address** }
- enum **pathType** { **ABS_PATH** , **REL_PATH** , **OPAQUE_PART** }
- enum **uriType** { **ABSOLUTE** , **RELATIVE** }

Functions

- int replace_escaped (char *in, size_t index, size_t *max)

  Replaces an escaped sequence with its unescaped version as in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs)
- int copy_URL_list (URL_list *in, URL_list *out)

  Copies one URL_list into another.
- void free_URL_list (URL_list *list)

  Frees the memory associated with a URL_list.
- void print_uri (uri_type *in)

  Function useful in debugging for printing a parsed uri.
- void print_token (token *in)

  Function useful in debugging for printing a token.
- int token_string_casecmp (token *in1, const char *in2)

  Compares buffer in the token object with the buffer in in2.
- int token_cmp (token *in1, token *in2)

  Compares two tokens.
- int remove_escaped_chars (char *in, size_t *size)

  Removes http escaped characters such as: "%20" and replaces them with their character representation. i.e. "hello%20foo" -> "hello foo".
- int remove_dots (char *in, size_t size)

  Removes ".", and ".." from a path.
- char * resolve_rel_url (char *base_url, char *rel_url)

  resolves a relative url with a base url returning a NEW (dynamically allocated with malloc) full url.
- int parse_uri (const char *in, size_t max, uri_type *out)

  Parses a uri as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).
- int parse_token (char *in, token *out, int max_size)

9.137.1 Macro Definition Documentation

9.137.1.1 RESERVED

```
#define RESERVED ";/?:@&=+$,{}"
```

added {} for compatibility

9.137.2 Typedef Documentation

9.137.2.1 hostport_type

```
typedef struct HOSTPORT hostport_type
```

Represents a host port: e.g. "127.127.0.1:80" text is a token pointing to the full string representation.

9.137.2.2 token

```
typedef struct TOKEN token
```

Buffer used in parsinghttp messages, urls, etc. generally this simply holds a pointer into a larger array.

9.137.2.3 uri_type

```
typedef struct URI uri_type
```

Represents a URI used in parse_uri and elsewhere.

9.137.2.4 URL_list

```
typedef struct URL_LIST URL_list
```

Represents a list of URLs as in the "callback" header of SUBSCRIBE message in GENA. "char *" URLs holds dynamic memory.

9.137.3 Function Documentation

9.137.3.1 copy_URL_list()

```
int copy_URL_list (
            URL_list * in,
            URL_list * out )
```

Copies one URL_list into another.

This includes dynamically allocating the out->URLs field (the full string), and the structures used to hold the parsedURLs. This memory MUST be freed by the caller through: free_URL_list(&out).

Returns

- HTTP_SUCCESS - On Success.
- UPNP_E_OUTOF_MEMORY - On Failure to allocate memory.

Parameters

in	*in*	Source URL list.
out	*out*	Destination URL list.

References copy_token(), HOSTPORT::text, UPNP_E_OUTOF_MEMORY, and URL_LIST::URLs.

Referenced by copy_subscription().

9.137.3.2 free_URL_list()

```
void free_URL_list (
            URL_list * list )
```

Frees the memory associated with a URL_list.

Frees the dynamically allocated members of of list. Does NOT free the pointer to the list itself (i.e. does NOT free(list)).

Parameters

in	*list*	URL list object.

References URL_LIST::URLs.

Referenced by freeSubscription().

9.137.3.3 parse_token()

```
int parse_token (
            char * in,
            token * out,
            int max_size )
```

Returns

Parameters

in	*in*	.
out	*out*	.
in	*max_size*	.

9.137.3.4 parse_uri()

```
int parse_uri (
            const char * in,
```

```
            size_t max,
            uri_type * out )
```

Parses a uri as defined in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs).

Handles absolute, relative, and opaque uris. Parses into the following pieces: scheme, hostport, pathquery, fragment (path and query are treated as one token)

Caller should check for the pieces they require.

Returns

Parameters

in	*in*	Character string containing uri information to be parsed.
in	*max*	Maximum limit on the number of characters.
out	*out*	Output parameter which will have the parsed uri information.

References parse_hostport(), parse_scheme(), parse_uric(), and token_string_casecmp().

Referenced by create_url_list(), FindServiceControlURLPath(), FindServiceEventURLPath(), http_FixStrUrl(), and resolve_rel_url().

9.137.3.5 print_token()

```
void print_token (
            token * in )
```

Function useful in debugging for printing a token.

Parameters

in	*in*	Token object to print.

9.137.3.6 print_uri()

```
void print_uri (
            uri_type * in )
```

Function useful in debugging for printing a parsed uri.

Parameters

in	*in*	URI object to print.

9.137.3.7 remove_dots()

```
int remove_dots (
            char * in,
            size_t size )
```

Removes ".", and ".." from a path.

If a ".." can not be resolved (i.e. the .. would go past the root of the path) an error is returned.

The input IS modified in place.)

Note

Examples char path[30]="/../hello"; remove_dots(path, strlen(path)) -> UPNP_E_INVALID_URL char path[30]="/./hello"; remove_dots(path, strlen(path)) -> UPNP_E_SUCCESS, in = "/hello" char path[30]="/./hello/foo/../goodbye" -> UPNP_E_SUCCESS, in = "/hello/goodbye"

Returns

- UPNP_E_SUCCESS - On Success.
- UPNP_E_OUTOF_MEMORY - On failure to allocate memory.
- UPNP_E_INVALID_URL - Failure to resolve URL.

Parameters

in	*in*	String of characters from which "dots" have to be removed.
in	*size*	Size limit for the number of characters.

References UPNP_E_SUCCESS.

Referenced by process_request(), and resolve_rel_url().

9.137.3.8 remove_escaped_chars()

```
int remove_escaped_chars (
            char * in,
            size_t * size )
```

Removes http escaped characters such as: "%20" and replaces them with their character representation. i.e. "hello%20foo" -> "hello foo".

The input IS MODIFIED in place (shortened). Extra characters are replaced with **NULL**.

Returns

UPNP_E_SUCCESS.

Parameters

in,out	*in*	String of characters to be modified.
in,out	*size*	Size limit for the number of characters.

References replace_escaped(), and UPNP_E_SUCCESS.

Referenced by process_request().

9.137.3.9 replace_escaped()

```
int replace_escaped (
            char * in,
            size_t index,
            size_t * max )
```

Replaces an escaped sequence with its unescaped version as in http://www.ietf.org/rfc/rfc2396.txt (RFC explaining URIs)

Size of array is NOT checked (MUST be checked by caller)

Note

This function modifies the string. If the sequence is an escaped sequence it is replaced, the other characters in the string are shifted over, and NULL characters are placed at the end of the string.

Returns

Parameters

in,out	*in*	String of characters.
in	*index*	Index at which to start checking the characters.
out	*max*	.

Referenced by remove_escaped_chars().

9.137.3.10 resolve_rel_url()

```
char * resolve_rel_url (
            char * base_url,
            char * rel_url )
```

resolves a relative url with a base url returning a NEW (dynamically allocated with malloc) full url.

If the base_url is **NULL**, then a copy of the rel_url is passed back if the rel_url is absolute then a copy of the rel_url is passed back if neither the base nor the rel_url are Absolute then NULL is returned. Otherwise it tries and resolves the relative url with the base as described in http://www.ietf.org/rfc/rfc2396.txt (RFCs explaining URIs).

The resolution of '..' is NOT implemented, but '.' is resolved.

Returns

Parameters

in	*base_url*	Base URL.
in	*rel_url*	Relative URL.

References parse_uri(), remove_dots(), HOSTPORT::text, and UPNP_E_SUCCESS.

Referenced by UpnpResolveURL(), and UpnpResolveURL2().

9.137.3.11 token_cmp()

```
int token_cmp (
            token * in1,
            token * in2 )
```

Compares two tokens.

Returns

- $<$ 0, if string1 is less than string2.
- == 0, if string1 is identical to string2 .
- $>$ 0, if string1 is greater than string2.

Parameters

in	*in1*	First token object whose buffer is to be compared.
in	*in2*	Second token object used for the comparison.

Referenced by FindServiceControlURLPath(), and FindServiceEventURLPath().

9.137.3.12 token_string_casecmp()

```
int token_string_casecmp (
            token * in1,
            const char * in2 )
```

Compares buffer in the token object with the buffer in in2.

Returns

- $<$ 0, if string1 is less than string2.
- == 0, if string1 is identical to string2 .
- $>$ 0, if string1 is greater than string2.

Parameters

in	*in1*	Token object whose buffer is to be compared.
in	*in2*	String of characters to compare with.

Referenced by http_FixUrl(), http_OpenHttpConnection(), and parse_uri().

9.138 uri.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENLIB_NET_URI_H
00033 #define GENLIB_NET_URI_H
00034
00039 #if !defined(_WIN32)
00040         #include <sys/param.h>
00041 #endif
00042
00043 #include "UpnpGlobal.h" /* for */
00044 #include "UpnpInet.h"
00045
00046 #include <ctype.h>
00047 #include <errno.h>
00048 #include <fcntl.h>
00049 #include <stdlib.h>
00050 #include <string.h>
00051 #include <sys/types.h>
00052 #include <time.h>
00053
00054 #ifdef _WIN32
00055         #if !defined(UPNP_USE_MSVCPP) && !defined(UPNP_USE_BCBPP)
00056                 /* VC Winsocks2 includes these functions */
00057                 #include "inet_pton.h"
00058         #endif
00059 #else
00060         #include <netdb.h> /* for struct addrinfo */
00061 #endif
00062
00063 #ifdef _WIN32
00064         #define strncasecmp strnicmp
00065 #else
00066         /* Other systems have strncasecmp */
00067 #endif
00068
00069 #ifdef __cplusplus
00070 extern "C" {
00071 #endif
00072
00073 #define MARK "-_.!~*'()"
00074
00076 #define RESERVED ";/?:@&=+$,{}"
00077
00078 #define HTTP_SUCCESS 1
00079
00080 enum hostType
00081 {
```

```
00082         HOSTNAME,
00083         IPv4address
00084 };
00085
00086 enum pathType
00087 {
00088         ABS_PATH,
00089         REL_PATH,
00090         OPAQUE_PART
00091 };
00092
00093 #ifdef _WIN32
00094 /* there is a conflict in windows with other symbols. */
00095 enum uriType
00096 {
00097         absolute,
00098         relative
00099 };
00100 #else
00101 enum uriType
00102 {
00103         ABSOLUTE,
00104         RELATIVE
00105 };
00106 #endif
00107
00112 typedef struct TOKEN
00113 {
00114         const char *buff;
00115         size_t size;
00116 } token;
00117
00122 typedef struct HOSTPORT
00123 {
00125         token text;
00126         /* Network Byte Order */
00127         struct sockaddr_storage IPaddress;
00128 } hostport_type;
00129
00133 typedef struct URI
00134 {
00135         enum uriType type;
00136         token scheme;
00137         enum pathType path_type;
00138         token pathquery;
00139         token fragment;
00140         hostport_type hostport;
00141 } uri_type;
00142
00147 typedef struct URL_LIST
00148 {
00150         size_t size;
00152         char *URLs;
00154         uri_type *parsedURLs;
00155 } URL_list;
00156
00169 int replace_escaped(
00171         char *in,
00173         size_t index,
00175         size_t *max);
00176
00188 int copy_URL_list(
00190         URL_list *in,
00192         URL_list *out);
00193
00200 void free_URL_list(
00202         URL_list *list);
00203
00207 #ifdef DEBUG
00208 void print_uri(
00210         uri_type *in);
00211 #else
00212         #define print_uri(in) \
00213                 do { \
00214                 } while (0)
00215 #endif
00216
00220 #ifdef DEBUG
00221 void print_token(
00223         token *in);
00224 #else
00225         #define print_token(in) \
00226                 do { \
00227                 } while (0)
00228 #endif
00229
00238 int token_string_casecmp(
```

```
00240         token *in1,
00242         const char *in2);
00243
00252 int token_cmp(
00254         token *in1,
00256         token *in2);
00257
00267 int remove_escaped_chars(
00269         char *in,
00271         size_t *size);
00272
00295 int remove_dots(
00297         char *in,
00299         size_t size);
00300
00315 char *resolve_rel_url(
00317         char *base_url,
00319         char *rel_url);
00320
00333 int parse_uri(
00335         const char *in,
00337         size_t max,
00340         uri_type *out);
00341
00347 int parse_token(
00349         char *in,
00351         token *out,
00353         int max_size);
00354
00355 #ifdef __cplusplus
00356 }
00357 #endif
00358
00359 #endif /* GENLIB_NET_URI_H */
```

9.139 urlconfig.h

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef URLCONFIG_H
00033 #define URLCONFIG_H
00034
00035 #include "UpnpInet.h"
00036
00037 /* functions available only if the web server is included */
00038
00039 #ifdef __cplusplus
00040 extern "C" {
00041 #endif
00042
00043 /*******************************************************************************
00044  * Function: configure_urlbase
00045  *
00046  * Parameters :
```

```
00047  *      INOUT IXML_Document *doc ;      IXML Description document
00048  *      IN const struct sockaddr *serverAddr;   socket address object
00049  *              providing the IP address and port information
00050  *      IN const char* alias ;          string containing the alias
00051  *      IN time_t last_modified ;       time when the XML document was
00052  *              downloaded
00053  *      OUT char docURL[LINE_SIZE] ;    buffer to hold the URL of the
00054  *              document.
00055  *
00056  * Description : Configure the full URL for the description document.
00057  *      Create the URL document and add alias, description information.
00058  *      The doc is added to the web server to be served using the given
00059  *      alias.
00060  *
00061  * Return : int ;
00062  *      UPNP_E_SUCCESS - On Success
00063  *      UPNP_E_OUTOF_MEMORY - Default Error
00064  ****************************************************************************/
00065 int configure_urlbase(IXML_Document *doc,
00066         const struct sockaddr *serverAddr,
00067         const char *alias,
00068         time_t last_modified,
00069         char docURL[LINE_SIZE]);
00070
00071 #ifdef __cplusplus
00072 } /* extern C */
00073 #endif
00074
00075 #endif /* URLCONFIG_H */
```

9.140 uuid.h

```
00001 #ifndef UUID_H
00002 #define UUID_H
00003
00004 /*
00005  * Copyright (c) 1990- 1993, 1996 Open Software Foundation, Inc.
00006  * Copyright (c) 1989 by Hewlett-Packard Company, Palo Alto, Ca. &
00007  * Digital Equipment Corporation, Maynard, Mass.
00008  * Copyright (c) 1998 Microsoft.
00009  * To anyone who acknowledges that this file is provided "AS IS"
00010  * without any express or implied warranty: permission to use, copy,
00011  * modify, and distribute this file for any purpose is hereby
00012  * granted without fee, provided that the above copyright notices and
00013  * this notice appears in all source code copies, and that none of
00014  * the names of Open Software Foundation, Inc., Hewlett-Packard
00015  * Company, or Digital Equipment Corporation be used in advertising
00016  * or publicity pertaining to distribution of the software without
00017  * specific, written prior permission.  Neither Open Software
00018  * Foundation, Inc., Hewlett-Packard Company, Microsoft, nor Digital Equipment
00019  * Corporation makes any representations about the suitability of
00020  * this software for any purpose.
00021  */
00022
00023 #include "sysdep.h"
00024
00026 typedef struct _uuid_upnp
00027 {
00029         uint32_t time_low;
00031         uint16_t time_mid;
00033         uint16_t time_hi_and_version;
00035         uint8_t clock_seq_hi_and_reserved;
00037         uint8_t clock_seq_low;
00039         uint8_t node[6];
00040 } uuid_upnp;
00041
00045 int uuid_create(
00047         uuid_upnp *id);
00048
00052 void upnp_uuid_unpack(
00054         uuid_upnp *u,
00056         char *out);
00057
00061 void uuid_create_from_name(
00063         uuid_upnp *uid,
00066         uuid_upnp nsid,
00068         void *name,
00070         int namelen);
00071
00082 int uuid_compare(
00084         uuid_upnp *u1,
00086         uuid_upnp *u2);
00087 #endif /* UUID_H */
```

9.141 VirtualDir.h

```
00001 #ifndef VIRTUALDIR_H
00002 #define VIRTUALDIR_H
00003
00004 #include "upnp.h"
00005
00010 struct VirtualDirCallbacks
00011 {
00014         VDCallback_GetInfo get_info;
00015
00019         VDCallback_Open open;
00020
00030         VDCallback_Read read;
00031
00037         VDCallback_Write write;
00038
00051         VDCallback_Seek seek;
00052
00057         VDCallback_Close close;
00058 };
00059
00060 typedef struct virtual_Dir_List
00061 {
00062         struct virtual_Dir_List *next;
00063         const void *cookie;
00064         char dirName[NAME_SIZE];
00065 } virtualDirList;
00066
00067 #endif /* VIRTUALDIR_H */
```

9.142 webserver.h

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * - Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * - Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * - Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef GENLIB_NET_HTTP_WEBSERVER_H
00033 #define GENLIB_NET_HTTP_WEBSERVER_H
00034
00035 #include "httpparser.h"
00036 #include "sock.h"
00037 #include <time.h>
00038
00039 #ifdef __cplusplus
00040 extern "C" {
00041 #endif
00042
00043 struct SendInstruction
00044 {
00045         int IsVirtualFile;
00046         int IsChunkActive;
00047         int IsRangeActive;
00048         int IsTrailers;
00049         char RangeHeader[200];
00050         char AcceptLanguageHeader[200];
```

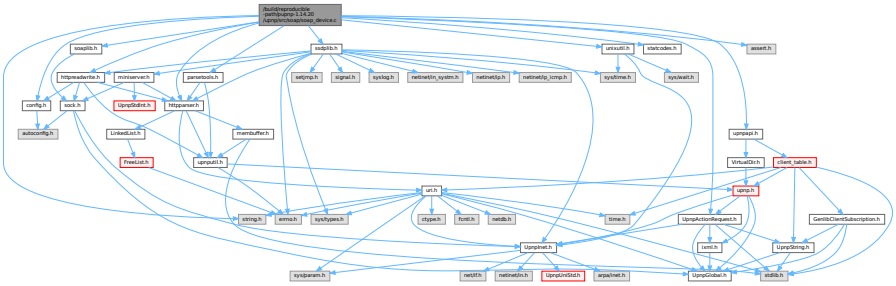
```
00051         off_t RangeOffset;
00053         off_t ReadSendSize;
00055         long RecvWriteSize;
00057         const void *CorsHeader;
00059         const void *Cookie;
00061         const void *RequestCookie;
00062         /* Later few more member could be added depending
00063          * on the requirement.*/
00064 };
00065
00077 int web_server_init(void);
00078
00084 void web_server_destroy(void);
00085
00096 int web_server_set_alias(
00099         const char *alias_name,
00102         const char *alias_content,
00104         size_t alias_content_length,
00107         time_t last_modified);
00108
00116 int web_server_set_root_dir(
00118         const char *root_dir);
00119
00126 int web_server_set_cors(
00128         const char *cors_string);
00129
00134 void web_server_callback(
00136         http_parser_t *parser,
00138         http_message_t *req,
00140         SOCKINFO *info);
00141
00142 #ifdef __cplusplus
00143 } /* extern C */
00144 #endif
00145
00146 #endif /* GENLIB_NET_HTTP_WEBSERVER_H */
```

9.143 /build/reproducible-path/pupnp-1.14.20/upnp/src/inet_pton.c File Reference

9.144 /build/reproducible-path/pupnp-1.14.20/upnp/src/soap/soap_↩ device.c File Reference

```
#include "config.h"
#include "UpnpActionRequest.h"
#include "httpparser.h"
#include "httpreadwrite.h"
#include "parsetools.h"
#include "soaplib.h"
#include "ssdplib.h"
#include "statcodes.h"
#include "unixutil.h"
#include "upnpapi.h"
#include <assert.h>
#include <string.h>
```

Include dependency graph for soap_device.c:



Data Structures

- struct soap_devserv_t

Macros

- #define SOAP_TIMEOUT UPNP_TIMEOUT
- #define **SREQ_HDR_NOT_FOUND** -1
- #define **SREQ_BAD_HDR_FORMAT** -2
- #define **SREQ_NOT_EXTENDED** -3
- #define **SOAP_INVALID_ACTION** 401
- #define **SOAP_INVALID_ARGS** 402
- #define **SOAP_OUT_OF_SYNC** 403
- #define **SOAP_INVALID_VAR** 404
- #define **SOAP_ACTION_FAILED** 501
- #define **SOAP_MEMORY_OUT** 603

Typedefs

- typedef struct soap_devserv_t **soap_devserv_t**

Functions

- static void send_error_response (SOCKINFO *info, int error_code, const char *err_msg, http_message_t *hmsg)

  Sends SOAP error response.
- static UPNP_INLINE void send_var_query_response (SOCKINFO *info, const char *var_value, http_message_t *hmsg)

  Sends response of get var status.
- static UPNP_INLINE void send_action_response (SOCKINFO *info, IXML_Document *action_resp, http_message_t *request)

  Sends the SOAP action response.
- static UPNP_INLINE void handle_query_variable (SOCKINFO *info, http_message_t *request, soap_devserv_t *soap_info, IXML_Node *req_node)

  Handles the SOAP requests to querry the state variables. This functionality has been deprecated in the UPnP V1.0 architecture.
- static void handle_invoke_action (SOCKINFO *info, http_message_t *request, soap_devserv_t *soap_info, IXML_Node *req_node)

Handles the SOAP action request.

- static int get_dev_service (http_message_t *request, int AddressFamily, soap_devserv_t *soap_info)

  Retrieve SOAP device/service information associated with request-URI, which includes the callback function to hand-over the request to the device application.
- static int get_mpost_acton_hdrval (http_message_t *request, memptr *val)

  Get the SOAPACTION header value for M-POST request.
- static int check_soapaction_hdr (http_message_t *request, soap_devserv_t *soap_info)

  Check the header validity, and get the action name and the version of the service that the CP wants to use.
- static int check_soap_request (soap_devserv_t *soap_info, IXML_Document *xml_doc, IXML_Node **req_node)

  Check validity of the SOAP request per UPnP specification.
- void soap_device_callback (http_parser_t *parser, http_message_t *request, SOCKINFO *info)

  This is a callback called by minisever after receiving the request from the control point. After HTTP processing, it calls handle_soap_request to start SOAP processing.

Variables

- static const char * **SOAP_BODY** = "Body"
- static const char * SOAP_URN
- static const char * **QUERY_STATE_VAR_URN** = "urn:schemas-upnp-org:control-1-0"
- static const char * **Soap_Invalid_Action** = "Invalid Action"
- static const char * **Soap_Action_Failed** = "Action Failed"
- static const char * **Soap_Invalid_Var** = "Invalid Var"
- static const char * **Soap_Memory_out** = "Out of Memory"

9.144.1 Macro Definition Documentation

9.144.1.1 SOAP_TIMEOUT

```
#define SOAP_TIMEOUT UPNP_TIMEOUT
```

timeout duration in secs for transmission/reception

9.144.2 Function Documentation

9.144.2.1 check_soap_request()

```
static int check_soap_request (
            soap_devserv_t * soap_info,
            IXML_Document * xml_doc,
            IXML_Node ** req_node )  [static]
```

Check validity of the SOAP request per UPnP specification.

Returns

0 if OK, -1 on failure.

Parameters

in	*soap_info*	SOAP device/service information.
in	*xml_doc*	Document containing the SOAP action request.
out	*req_node*	Node containing the SOAP action request/variable name.

References memptr::buf, DOMString, ixmlNode_getFirstChild(), ixmlNode_getLocalName(), ixmlNode_getNamespaceURI(), and ixmlNode_getNodeType().

Referenced by soap_device_callback().

9.144.2.2 check_soapaction_hdr()

```
static int check_soapaction_hdr (
            http_message_t * request,
            soap_devserv_t * soap_info )  [static]
```

Check the header validity, and get the action name and the version of the service that the CP wants to use.

Returns

UPNP_E_SUCCESS if OK, error number on failure.

Parameters

in	*request*	HTTP request.
in,out	*soap_info*	SOAP device/service information.

References memptr::buf, get_mpost_acton_hdrval(), memptr::length, http_message_t::method, namecopy(), PARSE_OK, and UPNP_E_SUCCESS.

Referenced by soap_device_callback().

9.144.2.3 get_dev_service()

```
static int get_dev_service (
            http_message_t * request,
            int AddressFamily,
            soap_devserv_t * soap_info )  [static]
```

Retrieve SOAP device/service information associated with request-URI, which includes the callback function to hand-over the request to the device application.

Returns

0 if OK, -1 on error.

Parameters

in	*request*	HTTP request.
in	*AddressFamily*	Address family: AF_INET or AF_INET6.
out	*soap_info*	SOAP device/service information.

References Handle_Info::Callback, Handle_Info::Cookie, GetDeviceHandleInfoForPath(), namecopy(), and http_message_t::uri.

Referenced by soap_device_callback().

9.144.2.4 get_mpost_acton_hdrval()

```
static int get_mpost_acton_hdrval (
            http_message_t * request,
            memptr * val )  [static]
```

Get the SOAPACTION header value for M-POST request.

Returns

UPNP_E_SUCCESS if OK, error number on failure.

Parameters

in	*request*	HTTP request.
out	*val*	Buffer to get the header value

References memptr::buf, membuffer::buf, memptr::length, membuffer::length, http_message_t::method, PARSE_OK, UPNP_E_OUTOF_MEMORY, UPNP_E_SUCCESS, and http_header_t::value.

Referenced by check_soapaction_hdr().

9.144.2.5 handle_invoke_action()

```
static void handle_invoke_action (
            SOCKINFO * info,
            http_message_t * request,
            soap_devserv_t * soap_info,
            IXML_Node * req_node )  [static]
```

Handles the SOAP action request.

Parameters

in	*info*	Socket info.
in	*request*	HTTP Request.
in	*soap_info*	SOAP device/service information.
in	*req_node*	Node containing the SOAP action request.

References memptr::buf, DOMString, SOCKINFO::foreign_sockaddr, ixmlDocument_free(), ixmlFreeDOMString(), ixmlParseBufferEx(), ixmlPrintNode(), memptr::length, send_action_response(), send_error_response(), UPNP_CONTROL_ACTION_REQUEST, UPNP_E_SUCCESS, UpnpActionRequest_delete(), UpnpActionRequest_get_ActionResult UpnpActionRequest_get_ErrCode(), UpnpActionRequest_get_ErrStr_cstr(), UpnpActionRequest_new(), UpnpActionRequest_set_Ac UpnpActionRequest_set_ActionResult(), UpnpActionRequest_set_CtrlPtIPAddr(), UpnpActionRequest_set_ErrCode(), UpnpActionRequest_strcpy_ActionName(), UpnpActionRequest_strcpy_DevUDN(), UpnpActionRequest_strcpy_ServiceID(), UpnpActionRequest_strncpy_Os(), and UpnpPrintf().

Referenced by soap_device_callback().

9.144.2.6 handle_query_variable()

```
static UPNP_INLINE void handle_query_variable (
             SOCKINFO * info,
             http_message_t * request,
             soap_devserv_t * soap_info,
             IXML_Node * req_node )  [static]
```

Handles the SOAP requests to querry the state variables. This functionality has been deprecated in the UPnP V1.0 architecture.

Parameters

in	*info*	Socket info.
in	*request*	HTTP Request.
in	*soap_info*	SOAP device/service information.
in	*req_node*	Node containing variable name.

References DOMString, SOCKINFO::foreign_sockaddr, ixmlNode_getNodeValue(), send_error_response(), send_var_query_response(), UPNP_CONTROL_GET_VAR_REQUEST, UPNP_E_SUCCESS, UpnpPrintf(), UpnpStateVarRequest_delete(), UpnpStateVarRequest_get_CurrentVal(), UpnpStateVarRequest_get_ErrCode(), UpnpStateVarRequest_get_ErrStr(), UpnpStateVarRequest_get_ErrStr_cstr(), UpnpStateVarRequest_new(), UpnpStateVarRequest_set_CtrlPtIPAddr(), UpnpStateVarRequest_set_ErrCode(), UpnpStateVarRequest_strcpy_DevUDN(), UpnpStateVarRequest_strcpy_ServiceID(), UpnpStateVarRequest_strcpy_StateVarName(), and UpnpString_get_Length().

Referenced by soap_device_callback().

9.144.2.7 send_action_response()

```
static UPNP_INLINE void send_action_response (
             SOCKINFO * info,
             IXML_Document * action_resp,
             http_message_t * request )  [static]
```

Sends the SOAP action response.

Parameters

in	*info*	Socket info.
in	*action_resp*	The response document.
in	*request*	Action request document.

References membuffer::buf, http_MakeMessage(), http_SendMessage(), ixmlFreeDOMString(), ixmlPrintNode(), membuffer::length, send_error_response(), SOAP_TIMEOUT, UPNP_E_OUTOF_MEMORY, and UpnpPrintf().

Referenced by handle_invoke_action().

9.144.2.8 send_error_response()

```
static void send_error_response (
            SOCKINFO * info,
            int error_code,
            const char * err_msg,
            http_message_t * hmsg )  [static]
```

Sends SOAP error response.

Parameters

in	*info*	Socket info.
in	*error_code*	Error code.
in	*err_msg*	Error message.
in	*hmsg*	HTTP request.

References membuffer::buf, http_MakeMessage(), http_SendMessage(), membuffer::length, and SOAP_TIMEOUT.

Referenced by handle_invoke_action(), handle_query_variable(), and send_action_response().

9.144.2.9 send_var_query_response()

```
static UPNP_INLINE void send_var_query_response (
            SOCKINFO * info,
            const char * var_value,
            http_message_t * hmsg )  [static]
```

Sends response of get var status.

Parameters

in	*info*	Socket info.
in	*var_value*	Value of the state variable.
in	*hmsg*	HTTP request.

References membuffer::buf, http_MakeMessage(), http_SendMessage(), membuffer::length, and SOAP_TIMEOUT.

Referenced by handle_query_variable().

9.144.2.10 soap_device_callback()

```
void soap_device_callback (
            http_parser_t * parser,
```

```
        http_message_t * request,
        SOCKINFO * info )
```

This is a callback called by minisever after receiving the request from the control point. After HTTP processing, it calls handle_soap_request to start SOAP processing.

This is a callback called by minisever after receiving the request from the control point. This function will start processing the request. It calls handle_invoke_action to handle the SOAP action.

Parameters

in	*parser*	Parsed request received by the device.
in	*request*	HTTP request.
in,out	*info*	Socket info.

References memptr::buf, check_soap_request(), check_soapaction_hdr(), http_message_t::entity, SOCKINFO::foreign_sockaddr, get_dev_service(), handle_invoke_action(), handle_query_variable(), has_xml_content_type(), ixmlDocument_free(), ixmlParseBufferEx(), UPNP_E_OUTOF_MEMORY, and UPNP_E_SUCCESS.

Referenced by UpnpInitPreamble().

9.144.3 Variable Documentation

9.144.3.1 SOAP_URN

```
const char* SOAP_URN  [static]
```

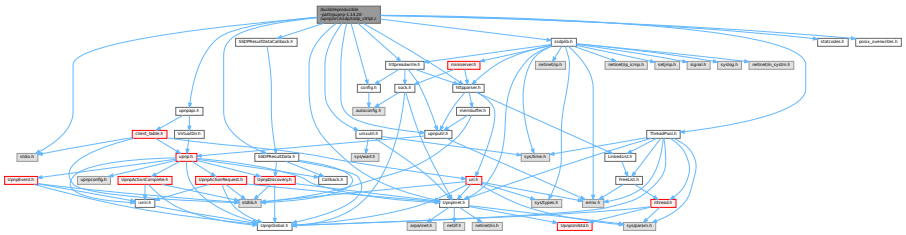
Initial value:

```
= "http:/"
                          "/schemas.xmlsoap.org/soap/envelope/"
```

9.145 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_↩ ctrlpt.c File Reference

```
#include "config.h"
#include "upnputil.h"
#include "SSDPResultData.h"
#include "SSDPResultDataCallback.h"
#include "ThreadPool.h"
#include "UpnpInet.h"
#include "httpparser.h"
#include "httpreadwrite.h"
#include "ssdplib.h"
#include "statcodes.h"
#include "unixutil.h"
#include "upnpapi.h"
#include <stdio.h>
```

```
#include "posix_overwrites.h"
```
Include dependency graph for ssdp_ctrlpt.c:



Functions

- static void send_search_result (void *data)

  Sends a callback to the control point application with a SEARCH result.
- static int CreateClientRequestPacket (char *RqstBuf, size_t RqstBufSize, int Mx, char *SearchTarget, int AddressFamily)

  Creates a HTTP search request packet depending on the input parameter.
- static void **searchExpired** (void *arg)

SSDP Control Point Functions

- void ssdp_handle_ctrlpt_msg (http_message_t *hmsg, struct sockaddr_storage *dest_addr, int timeout)

  This function handles the ssdp messages from the devices. These messages includes the search replies, advertisement of device coming alive and bye byes.
- int SearchByTarget (int Hnd, int Mx, char *St, void *Cookie)

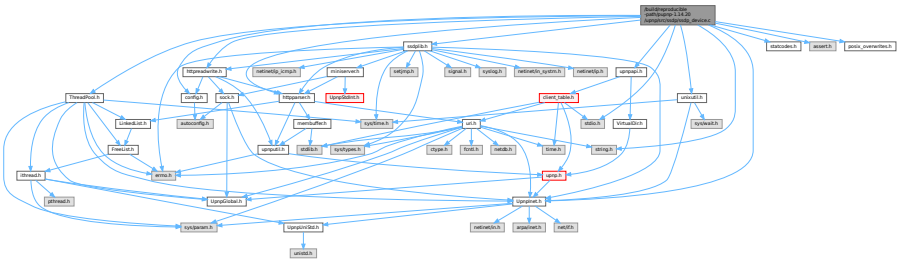
  Creates and send the search request for a specific URL.

9.146 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_↩ device.c File Reference

```
#include "config.h"
#include "ThreadPool.h"
#include "UpnpInet.h"
#include "httpparser.h"
#include "httpreadwrite.h"
#include "ssdplib.h"
#include "statcodes.h"
#include "unixutil.h"
#include "upnpapi.h"
#include <assert.h>
#include <stdio.h>
#include <string.h>
```

```
#include "posix_overwrites.h"
```

Include dependency graph for ssdp_device.c:



Macros

- #define **MSGTYPE_SHUTDOWN** 0
- #define **MSGTYPE_ADVERTISEMENT** 1
- #define **MSGTYPE_REPLY** 2
- #define **MX_FUDGE_FACTOR** 10

Functions

- static int NewRequestHandler (struct sockaddr *DestAddr, int NumPacket, char **RqPacket)

  Works as a request handler which passes the HTTP request string to multicast channel.
- static int extractIPv6address (char *url, char *address)
- static int isUrlV6UlaGua (char *descdocUrl)
- static void CreateServicePacket (int msg_type, const char *nt, char *usn, char *location, int duration, char **packet, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates a HTTP request packet. Depending on the input parameter, it either creates a service advertisement request or service shutdown request etc.

SSDP Device Functions

- void advertiseAndReplyThread (void *data)

  Wrapper function to reply the search request coming from the control point.
- void ssdp_handle_device_request (http_message_t *hmsg, struct sockaddr_storage *dest_addr)

  Handles the search request. It does the sanity checks of the request and then schedules a thread to send a random time reply (random within maximum time given by the control point to reply).
- int DeviceAdvertisement (char *DevType, int RootDev, char *Udn, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the device advertisement request based on the input parameter, and send it to the multicast channel.
- int SendReply (struct sockaddr *DestAddr, char *DevType, int RootDev, char *Udn, char *Location, int Duration, int ByType, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the reply packet based on the input parameter, and send it to the client addesss given in its input parameter DestAddr.
- int DeviceReply (struct sockaddr *DestAddr, char *DevType, int RootDev, char *Udn, char *Location, int Duration, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the reply packet based on the input parameter, and send it to the client address given in its input parameter DestAddr.
- int ServiceAdvertisement (char *Udn, char *ServType, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

  Creates the advertisement packet based on the input parameter, and send it to the multicast channel.
- int ServiceReply (struct sockaddr *DestAddr, char *ServType, char *Udn, char *Location, int Duration, int PowerState, int SleepPeriod, int RegistrationState)

Creates the advertisement packet based on the input parameter, and send it to the multicast channel.

- int ServiceShutdown (char *Udn, char *ServType, char *Location, int Duration, int AddressFamily, int PowerState, int SleepPeriod, int RegistrationState)

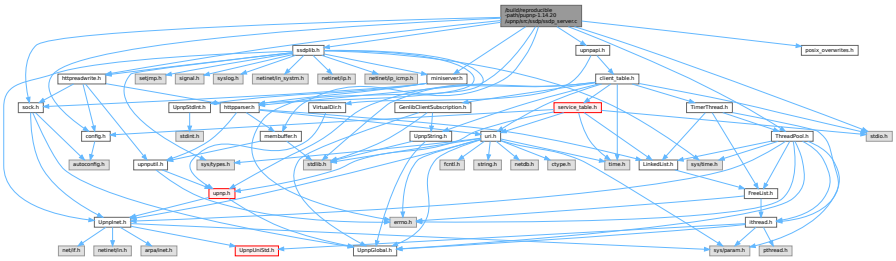
  Creates a HTTP service shutdown request packet and sends it to the multicast channel through RequestHandler.
- int DeviceShutdown (char *DevType, int RootDev, char *Udn, char *Location, int Duration, int Address↩ Family, int PowerState, int SleepPeriod, int RegistrationState)

  Creates a HTTP device shutdown request packet and send it to the multicast channel through RequestHandler.

9.147 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_↩ server.c File Reference

```
#include "config.h"
#include "ssdplib.h"
#include "ThreadPool.h"
#include "httpparser.h"
#include "httpreadwrite.h"
#include "membuffer.h"
#include "miniserver.h"
#include "sock.h"
#include "upnpapi.h"
#include <stdio.h>
#include "posix_overwrites.h"
```

Include dependency graph for ssdp_server.c:



Data Structures

- struct SSDPSockArray

Macros

- #define **MAX_TIME_TOREAD** 45

Enumerations

- enum **Listener** { **Idle** , **Stopping** , **Running** }

Functions

- void **RequestHandler** ()
- static void free_ssdp_event_handler_data (void *the_data)

  Frees the ssdp request.
- static UPNP_INLINE int valid_ssdp_msg (http_message_t *hmsg)

  Does some quick checking of the ssdp msg.
- static UPNP_INLINE int start_event_handler (void *Data)

  Parses the message and dispatches it to a handler which handles the ssdp request msg.
- static void ssdp_event_handler_thread (void *the_data)

  This function is a thread that handles SSDP requests.
- static int create_ssdp_sock_v4 (SOCKET *ssdpSock)
- static int create_ssdp_sock_reqv4 (SOCKET *ssdpReqSock)

  Creates the SSDP IPv4 socket to be used by the control point.

SSDP Server Functions

- int AdvertiseAndReply (int AdFlag, UpnpDevice_Handle Hnd, enum SsdpSearchType SearchType, struct sockaddr *DestAddr, char *DeviceType, char *DeviceUDN, char *ServiceType, int Exp)

  Sends SSDP advertisements, replies and shutdown messages.
- int unique_service_name (char *cmd, SsdpEvent *Evt)

  Fills the fields of the event structure like DeviceType, Device UDN and Service Type.
- enum SsdpSearchType ssdp_request_type1 (char *cmd)

  This function figures out the type of the SSDP search in the in the request.
- int ssdp_request_type (char *cmd, SsdpEvent *Evt)

  Starts filling the SSDP event structure based upon the request received.
- int readFromSSDPSocket (SOCKET socket)

  This function reads the data from the ssdp socket.
- int get_ssdp_sockets (MiniServerSockArray *out)

  Creates the IPv4 and IPv6 ssdp sockets required by the control point and device operation.

Variables

- SOCKET **gSsdpReqSocket4** = INVALID_SOCKET
- static const char **SERVICELIST_STR** [ ] = "serviceList"

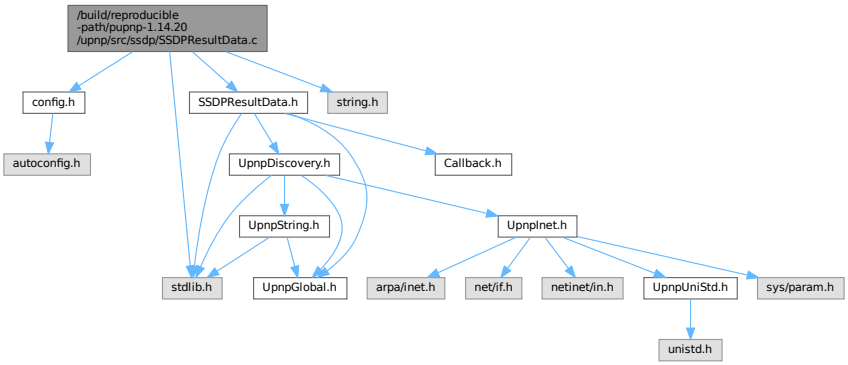
9.148 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/↩ SSDPResultData.c File Reference

Source file for SSDPResultData methods.

```
#include "config.h"
#include <stdlib.h>
#include <string.h>
```

```
#include "SSDPResultData.h"
```

Include dependency graph for SSDPResultData.c:



Data Structures

- struct s_SSDPResultData

Functions

- SSDPResultData * SSDPResultData_new ()
- void SSDPResultData_delete (SSDPResultData *q)
- int SSDPResultData_assign (SSDPResultData *p, const SSDPResultData *q)
- SSDPResultData * SSDPResultData_dup (const SSDPResultData *q)
- const UpnpDiscovery * SSDPResultData_get_Param (const SSDPResultData *p)
- int SSDPResultData_set_Param (SSDPResultData *p, const UpnpDiscovery *s)
- void * SSDPResultData_get_Cookie (const SSDPResultData *p)
- int SSDPResultData_set_Cookie (SSDPResultData *p, void *n)
- Upnp_FunPtr SSDPResultData_get_CtrlptCallback (const SSDPResultData *p)
- int SSDPResultData_set_CtrlptCallback (SSDPResultData *p, Upnp_FunPtr n)

9.148.1 Detailed Description

Source file for SSDPResultData methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.148.2 Function Documentation

9.148.2.1 SSDPResultData_assign()

```
int SSDPResultData_assign (
            SSDPResultData * p,
            const SSDPResultData * q )
```

Assignment operator

References SSDPResultData_get_Cookie(), SSDPResultData_get_CtrlptCallback(), SSDPResultData_get_Param(), SSDPResultData_set_Cookie(), SSDPResultData_set_CtrlptCallback(), and SSDPResultData_set_Param().

Referenced by SSDPResultData_dup().

9.148.2.2 SSDPResultData_delete()

```
void SSDPResultData_delete (
            SSDPResultData * p )
```

Destructor

References UpnpDiscovery_delete().

Referenced by send_search_result(), and ssdp_handle_ctrlpt_msg().

9.148.2.3 SSDPResultData_dup()

```
SSDPResultData * SSDPResultData_dup (
            const SSDPResultData * p )
```

Copy Constructor

References SSDPResultData_assign(), and SSDPResultData_new().

9.148.2.4 SSDPResultData_get_Cookie()

```
void * SSDPResultData_get_Cookie (
            const SSDPResultData * p )
```

SSDPResultData_get_Cookie

Referenced by SSDPResultData_assign().

9.148.2.5 SSDPResultData_get_CtrlptCallback()

```
Upnp_FunPtr SSDPResultData_get_CtrlptCallback (
            const SSDPResultData * p )
```

SSDPResultData_get_CtrlptCallback

Referenced by SSDPResultData_assign().

9.148.2.6 SSDPResultData_get_Param()

```
const UpnpDiscovery * SSDPResultData_get_Param (
            const SSDPResultData * p )
```

SSDPResultData_get_Param

Referenced by SSDPResultData_assign().

9.148.2.7 SSDPResultData_new()

```
SSDPResultData * SSDPResultData_new ( )
```

Constructor

References UpnpDiscovery_new().

Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_dup().

9.148.2.8 SSDPResultData_set_Cookie()

```
int SSDPResultData_set_Cookie (
            SSDPResultData * p,
            void * n )
```

SSDPResultData_set_Cookie

Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_assign().

9.148.2.9 SSDPResultData_set_CtrlptCallback()

```
int SSDPResultData_set_CtrlptCallback (
            SSDPResultData * p,
            Upnp_FunPtr n )
```

SSDPResultData_set_CtrlptCallback

Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_assign().

9.148.2.10 SSDPResultData_set_Param()

```
int SSDPResultData_set_Param (
            SSDPResultData * p,
            const UpnpDiscovery * n )
```

SSDPResultData_set_Param

References UpnpDiscovery_delete(), and UpnpDiscovery_dup().

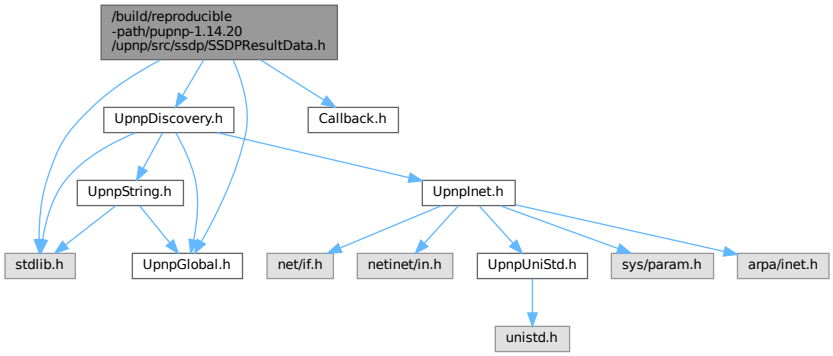
Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_assign().

9.149 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.h File Reference

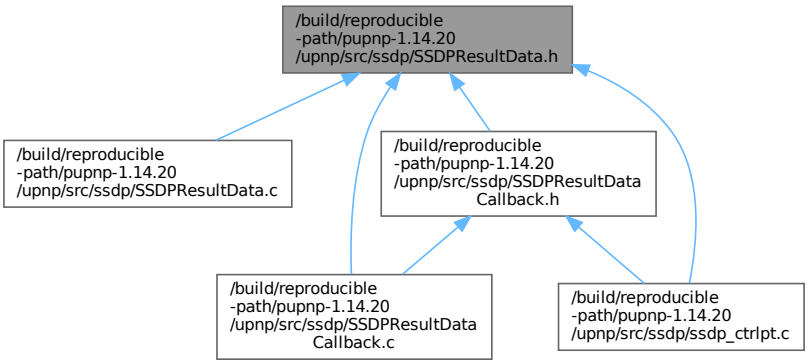
Header file for SSDPResultData methods.

```
#include <stdlib.h>
#include "UpnpGlobal.h"
#include "Callback.h"
#include "UpnpDiscovery.h"
```

Include dependency graph for SSDPResultData.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct s_SSDPResultData SSDPResultData

Functions

- UPNP_EXPORT_SPEC SSDPResultData * SSDPResultData_new ()
- UPNP_EXPORT_SPEC void SSDPResultData_delete (SSDPResultData *p)
- UPNP_EXPORT_SPEC SSDPResultData * SSDPResultData_dup (const SSDPResultData *p)
- UPNP_EXPORT_SPEC int SSDPResultData_assign (SSDPResultData *p, const SSDPResultData *q)
- UPNP_EXPORT_SPEC const UpnpDiscovery * SSDPResultData_get_Param (const SSDPResultData *p)
- UPNP_EXPORT_SPEC int SSDPResultData_set_Param (SSDPResultData *p, const UpnpDiscovery *n)
- UPNP_EXPORT_SPEC void * SSDPResultData_get_Cookie (const SSDPResultData *p)
- UPNP_EXPORT_SPEC int SSDPResultData_set_Cookie (SSDPResultData *p, void *n)
- UPNP_EXPORT_SPEC Upnp_FunPtr SSDPResultData_get_CtrlptCallback (const SSDPResultData *p)
- UPNP_EXPORT_SPEC int SSDPResultData_set_CtrlptCallback (SSDPResultData *p, Upnp_FunPtr n)

9.149.1 Detailed Description

Header file for SSDPResultData methods.

Do not edit this file, it is automatically generated. Please look at generator.c.

Author

Marcelo Roberto Jimenez

9.149.2 Typedef Documentation

9.149.2.1 SSDPResultData

```
typedef struct s_SSDPResultData SSDPResultData
```

SSDPResultData

9.149.3 Function Documentation

9.149.3.1 SSDPResultData_assign()

```
UPNP_EXPORT_SPEC int SSDPResultData_assign (
            SSDPResultData * p,
            const SSDPResultData * q )
```

Assignment operator

References SSDPResultData_get_Cookie(), SSDPResultData_get_CtrlptCallback(), SSDPResultData_get_Param(), SSDPResultData_set_Cookie(), SSDPResultData_set_CtrlptCallback(), and SSDPResultData_set_Param().

Referenced by SSDPResultData_dup().

9.149.3.2 SSDPResultData_delete()

```
UPNP_EXPORT_SPEC void SSDPResultData_delete (
            SSDPResultData * p )
```

Destructor

References UpnpDiscovery_delete().

Referenced by send_search_result(), and ssdp_handle_ctrlpt_msg().

9.149.3.3 SSDPResultData_dup()

```
UPNP_EXPORT_SPEC SSDPResultData * SSDPResultData_dup (
            const SSDPResultData * p )
```

Copy Constructor

References SSDPResultData_assign(), and SSDPResultData_new().

9.149.3.4 SSDPResultData_get_Cookie()

```
UPNP_EXPORT_SPEC void * SSDPResultData_get_Cookie (
            const SSDPResultData * p )
```

SSDPResultData_get_Cookie

Referenced by SSDPResultData_assign().

9.149.3.5 SSDPResultData_get_CtrlptCallback()

```
UPNP_EXPORT_SPEC Upnp_FunPtr SSDPResultData_get_CtrlptCallback (
            const SSDPResultData * p )
```

SSDPResultData_get_CtrlptCallback

Referenced by SSDPResultData_assign().

9.149.3.6 SSDPResultData_get_Param()

```
UPNP_EXPORT_SPEC const UpnpDiscovery * SSDPResultData_get_Param (
            const SSDPResultData * p )
```

SSDPResultData_get_Param

Referenced by SSDPResultData_assign().

9.149.3.7 SSDPResultData_new()

```
UPNP_EXPORT_SPEC SSDPResultData * SSDPResultData_new ( )
```

Constructor

References UpnpDiscovery_new().

Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_dup().

9.149.3.8 SSDPResultData_set_Cookie()

```
UPNP_EXPORT_SPEC int SSDPResultData_set_Cookie (
            SSDPResultData * p,
            void * n )
```

SSDPResultData_set_Cookie

Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_assign().

9.149.3.9 SSDPResultData_set_CtrlptCallback()

```
UPNP_EXPORT_SPEC int SSDPResultData_set_CtrlptCallback (
            SSDPResultData * p,
            Upnp_FunPtr n )
```

SSDPResultData_set_CtrlptCallback

Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_assign().

9.149.3.10 SSDPResultData_set_Param()

```
UPNP_EXPORT_SPEC int SSDPResultData_set_Param (
            SSDPResultData * p,
            const UpnpDiscovery * n )
```

SSDPResultData_set_Param

References UpnpDiscovery_delete(), and UpnpDiscovery_dup().

Referenced by ssdp_handle_ctrlpt_msg(), and SSDPResultData_assign().

9.150 SSDPResultData.h

Go to the documentation of this file.

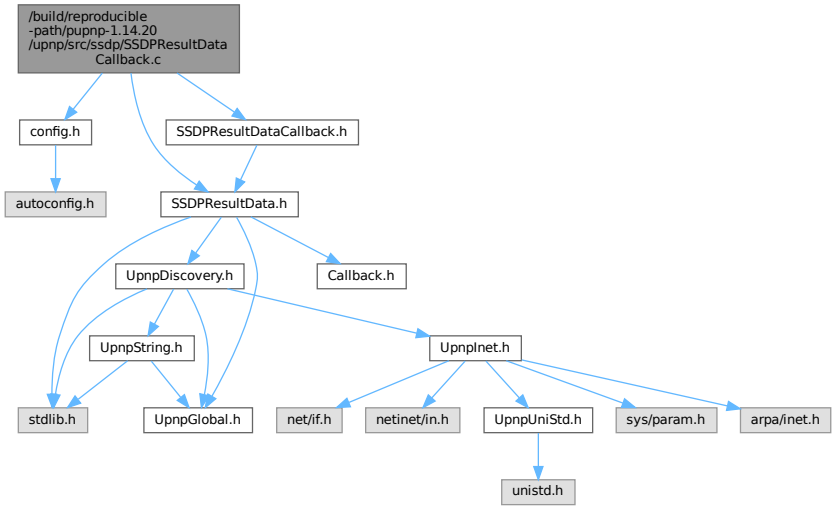
```
00001 #ifndef SSDPRESULTDATA_H
00002 #define SSDPRESULTDATA_H
00003
00014 #include <stdlib.h> /* for size_t */
00015
00016 #include "UpnpGlobal.h" /* for UPNP_EXPORT_SPEC */
00017
00018 #include "Callback.h"
00019 #include "UpnpDiscovery.h"
00020
00021 #ifdef __cplusplus
00022 extern "C" {
00023 #endif /* __cplusplus */
00024
00028 typedef struct s_SSDPResultData SSDPResultData;
00029
00031 UPNP_EXPORT_SPEC SSDPResultData *SSDPResultData_new();
00033 UPNP_EXPORT_SPEC void SSDPResultData_delete(SSDPResultData *p);
00035 UPNP_EXPORT_SPEC SSDPResultData *SSDPResultData_dup(const SSDPResultData *p);
00037 UPNP_EXPORT_SPEC int SSDPResultData_assign(
00038         SSDPResultData *p, const SSDPResultData *q);
00039
00041 UPNP_EXPORT_SPEC const UpnpDiscovery *SSDPResultData_get_Param(
00042         const SSDPResultData *p);
00044 UPNP_EXPORT_SPEC int SSDPResultData_set_Param(
00045         SSDPResultData *p, const UpnpDiscovery *n);
00046
00048 UPNP_EXPORT_SPEC void *SSDPResultData_get_Cookie(const SSDPResultData *p);
00050 UPNP_EXPORT_SPEC int SSDPResultData_set_Cookie(SSDPResultData *p, void *n);
00051
00053 UPNP_EXPORT_SPEC Upnp_FunPtr SSDPResultData_get_CtrlptCallback(
00054         const SSDPResultData *p);
00056 UPNP_EXPORT_SPEC int SSDPResultData_set_CtrlptCallback(
00057         SSDPResultData *p, Upnp_FunPtr n);
00058
00059 #ifdef __cplusplus
00060 }
00061 #endif /* __cplusplus */
00062
00063 #endif /* SSDPRESULTDATA_H */
```

9.151 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/↩ SSDPResultDataCallback.c File Reference

SSDPResultDataCallback.

```
#include "config.h"
#include "SSDPResultData.h"
#include "SSDPResultDataCallback.h"
```

Include dependency graph for SSDPResultDataCallback.c:



Functions

- void **SSDPResultData_Callback** (const SSDPResultData *p)

9.151.1 Detailed Description

SSDPResultDataCallback.

Author

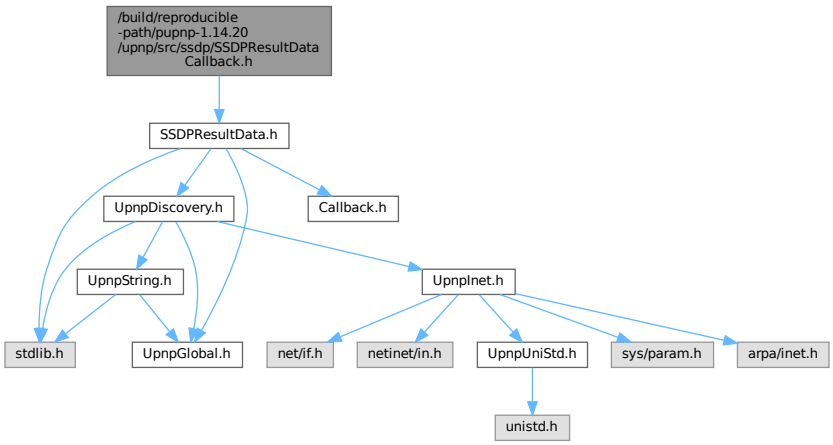
Marcelo Roberto Jimenez

9.152 /build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/↩ SSDPResultDataCallback.h File Reference

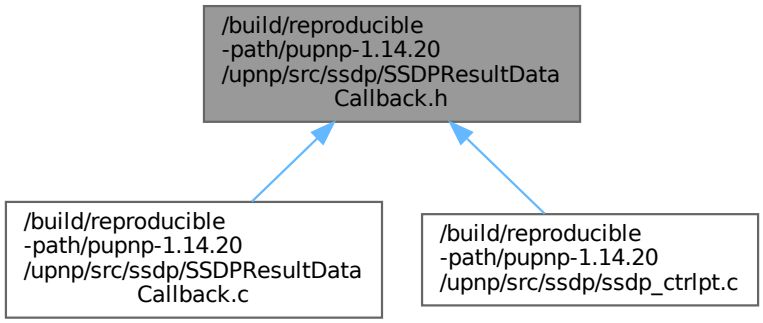
SSDPResultData object declararion.

```
#include "SSDPResultData.h"
```

Include dependency graph for SSDPResultDataCallback.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **SSDPResultData_Callback** (const SSDPResultData *p)

9.152.1 Detailed Description

SSDPResultData object declararion.

Author

Marcelo Roberto Jimenez

9.153 SSDPResultDataCallback.h

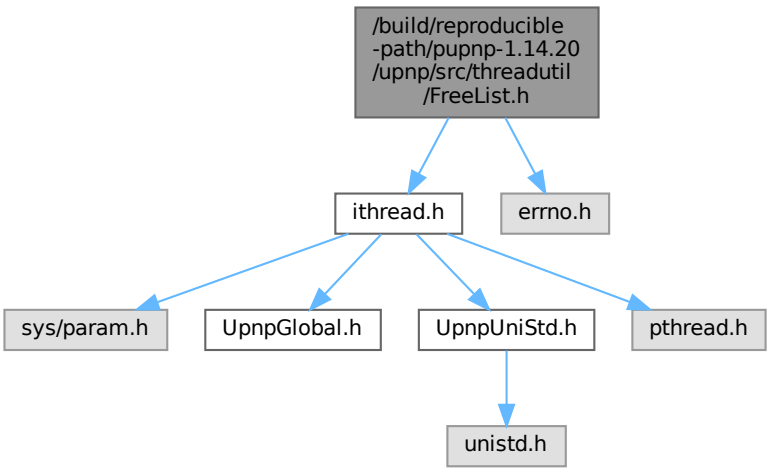
Go to the documentation of this file.

```
00001 #ifndef SSDPRESULTDATACALLBACK_H
00002 #define SSDPRESULTDATACALLBACK_H
00003
00004 #include "SSDPResultData.h"
00005
00019 void SSDPResultData_Callback(const SSDPResultData *p);
00020
00021 /* @} SSDPlib */
00022
00023 #endif /* SSDPRESULTDATACALLBACK_H */
```

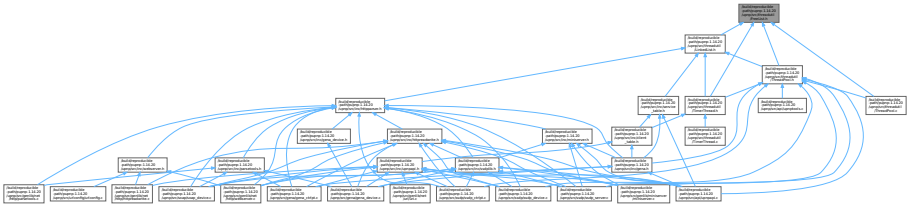
9.154 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/FreeList.h File Reference

```
#include "ithread.h"
#include <errno.h>
```

Include dependency graph for FreeList.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct FREELISTNODE
- struct FREELIST

Typedefs

- typedef struct FREELISTNODE FreeListNode
- typedef struct FREELIST FreeList

Functions

- int FreeListInit (FreeList *free_list, size_t elementSize, int maxFreeListLength)

  Initializes Free List.
- void * FreeListAlloc (FreeList *free_list)

  Allocates chunk of set size.
- int FreeListFree (FreeList *free_list, void *element)

  Returns an item to the Free List.
- int FreeListDestroy (FreeList *free_list)

  Releases the resources stored with the free list.

9.154.1 Typedef Documentation

9.154.1.1 FreeList

```
typedef struct FREELIST FreeList
```

Stores head and size of free list, as well as mutex for protection.

9.154.1.2 FreeListNode

```
typedef struct FREELISTNODE FreeListNode
```

Free list node. points to next free item. Memory for node is borrowed from allocated items.

9.154.2 Function Documentation

9.154.2.1 FreeListAlloc()

```
void * FreeListAlloc (
            FreeList * free_list )
```

Allocates chunk of set size.

If a free item is available in the list, returnes the stored item, otherwise calls the O.S. to allocate memory.

Returns

Non NULL on success. NULL on failure.

Parameters

free_list	Must be valid, non null, pointer to a linked list.

9.154.2.2 FreeListDestroy()

```
int FreeListDestroy (
            FreeList * free_list )
```

Releases the resources stored with the free list.

Returns

:

- 0 on success.
- EINVAL on failure.

Parameters

free_list	Must be valid, non null, pointer to a linked list.

9.154.2.3 FreeListFree()

```
int FreeListFree (
            FreeList * free_list,
            void * element )
```

Returns an item to the Free List.

If the free list is smaller than the max size then adds the item to the free list, otherwise returns the item to the O.S.

Returns

:

- 0 on success.
- EINVAL on failure.

Parameters

free_list	Must be valid, non null, pointer to a free list.
element	Must be a pointer allocated by FreeListAlloc.

9.154.2.4 FreeListInit()

```
int FreeListInit (
            FreeList * free_list,
```

```
            size_t elementSize,
            int maxFreeListLength )
```

Initializes Free List.

Must be called first and only once for FreeList.

Returns

:

- 0 on success.
- EINVAL on failure.

Parameters

free_list	Must be valid, non null, pointer to a linked list.
elementSize	Size of elements to store in free list.
maxFreeListLength	Max size that the free list can grow to before returning memory to O.S.

9.155 FreeList.h

Go to the documentation of this file.

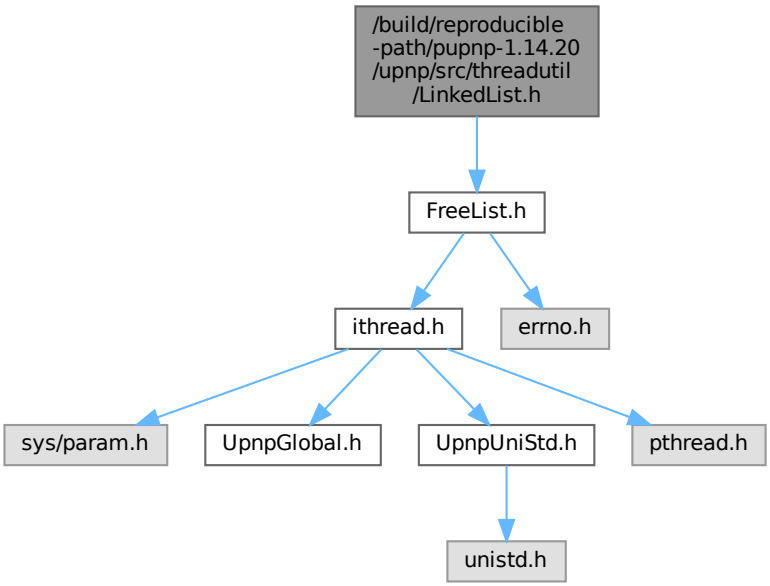
```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * * Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * * Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * * Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  *******************************************************************************/
00031
00032 #ifndef FREE_LIST_H
00033 #define FREE_LIST_H
00034
00039 #ifdef __cplusplus
00040 extern "C" {
00041 #endif
00042
00043 #include "ithread.h"
00044
00045 #include <errno.h>
00046
00052 typedef struct FREELISTNODE
00053 {
00054         struct FREELISTNODE *next;
```

```
00055 } FreeListNode;
00056
00061 typedef struct FREELIST
00062 {
00063         FreeListNode *head;
00064         size_t element_size;
00065         int maxFreeListLength;
00066         int freeListLength;
00067 } FreeList;
00068
00078 int FreeListInit(
00080         FreeList *free_list,
00082         size_t elementSize,
00085         int maxFreeListLength);
00086
00095 void *FreeListAlloc(
00097         FreeList *free_list);
00098
00109 int FreeListFree(
00111         FreeList *free_list,
00113         void *element);
00114
00122 int FreeListDestroy(
00124         FreeList *free_list);
00125
00126 #ifdef __cplusplus
00127 }
00128 #endif
00129
00130 #endif /* FREE_LIST_H */
```

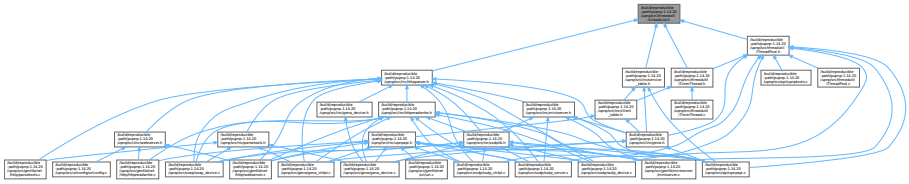
9.156 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/LinkedList.h File Reference

```
#include "FreeList.h"
```

Include dependency graph for LinkedList.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct LISTNODE
- struct LINKEDLIST

Macros

- #define **EOUTOFMEM** (-7 & 1 << 29)
- #define **FREELISTSIZE** 100
- #define **LIST_SUCCESS** 1
- #define **LIST_FAIL** 0

Typedefs

- typedef void(* free_function) (void *arg)
- typedef int(* cmp_routine) (void *itemA, void *itemB)
- typedef struct LISTNODE ListNode
- typedef struct LINKEDLIST LinkedList

Functions

- int ListInit (LinkedList *list, cmp_routine cmp_func, free_function free_func)

  Initializes LinkedList. Must be called first and only once for List.
- ListNode * ListAddHead (LinkedList *list, void *item)

  Adds a node to the head of the list. Node gets immediately after list head.
- ListNode * ListAddTail (LinkedList *list, void *item)

  Adds a node to the tail of the list. Node gets added immediately before list.tail.
- ListNode * ListAddAfter (LinkedList *list, void *item, ListNode *bnode)

  Adds a node after the specified node. Node gets added immediately after bnode.
- ListNode * ListAddBefore (LinkedList *list, void *item, ListNode *anode)

  Adds a node before the specified node. Node gets added immediately before anode.
- void * ListDelNode (LinkedList *list, ListNode *dnode, int freeItem)

  Removes a node from the list. The memory for the node is freed.
- int ListDestroy (LinkedList *list, int freeItem)

  *Removes all memory associated with list nodes. Does not free LinkedList *list.*
- ListNode * ListHead (LinkedList *list)

  Returns the head of the list.
- ListNode * ListTail (LinkedList *list)

  Returns the tail of the list.
- ListNode * ListNext (LinkedList *list, ListNode *node)

Returns the next item in the list.

- ListNode * ListPrev (LinkedList *list, ListNode *node)

  Returns the previous item in the list.

- ListNode * ListFind (LinkedList *list, ListNode *start, void *item)

  Finds the specified item in the list.

- long ListSize (LinkedList *list)

  Returns the size of the list.

9.156.1 Typedef Documentation

9.156.1.1 cmp_routine

```
typedef int(* cmp_routine) (void *itemA, void *itemB)
```

Function for comparing list items. Returns 1 if itemA==itemB

9.156.1.2 free_function

```
typedef void(* free_function) (void *arg)
```

Function for freeing list items.

9.156.1.3 LinkedList

```
typedef struct LINKEDLIST LinkedList
```

Linked list (no protection).

Because this is for internal use, parameters are NOT checked for validity. The first item of the list is stored at node: head->next The last item of the list is stored at node: tail->prev If head->next=tail, then list is empty. To iterate through the list:

```
LinkedList g;
ListNode *temp = NULL;
for (temp = ListHead(g);temp!=NULL;temp = ListNext(g,temp)) {
}
```

9.156.1.4 ListNode

```
typedef struct LISTNODE ListNode
```

Linked list node. Stores generic item and pointers to next and prev.

9.156.2 Function Documentation

9.156.2.1 ListAddAfter()

```
ListNode * ListAddAfter (
            LinkedList * list,
            void * item,
            ListNode * bnode )
```

Adds a node after the specified node. Node gets added immediately after bnode.

Precondition: The list has been initialized.

Returns

The pointer to the ListNode on success, NULL on failure.

Parameters

list	Must be valid, non null, pointer to a linked list.
item	Item to be added.
bnode	Node to add after.

References LINKEDLIST::size.

9.156.2.2 ListAddBefore()

```
ListNode * ListAddBefore (
            LinkedList * list,
            void * item,
            ListNode * anode )
```

Adds a node before the specified node. Node gets added immediately before anode.

Precondition: The list has been initialized.

Returns

The pointer to the ListNode on success, NULL on failure.

Parameters

list	Must be valid, non null, pointer to a linked list.
item	Item to be added.
anode	Node to add in front of.

References LINKEDLIST::size.

9.156.2.3 ListAddHead()

```
ListNode * ListAddHead (
            LinkedList * list,
            void * item )
```

Adds a node to the head of the list. Node gets immediately after list head.

Precondition: The list has been initialized.

Returns

The pointer to the ListNode on success, NULL on failure.

Parameters

list	Must be valid, non null, pointer to a linked list.
item	Item to be added.

References LINKEDLIST::head.

9.156.2.4 ListAddTail()

```
ListNode * ListAddTail (
            LinkedList * list,
            void * item )
```

Adds a node to the tail of the list. Node gets added immediately before list.tail.

Precondition: The list has been initialized.

Returns

The pointer to the ListNode on success, NULL on failure.

Parameters

list	Must be valid, non null, pointer to a linked list.
item	Item to be added.

References LINKEDLIST::tail.

9.156.2.5 ListDelNode()

```
void * ListDelNode (
            LinkedList * list,
            ListNode * dnode,
            int freeItem )
```

Removes a node from the list. The memory for the node is freed.

Precondition: The list has been initialized.

Returns

The pointer to the item stored in the node or NULL if the item is freed.

Parameters

list	Must be valid, non null, pointer to a linked list.
dnode	Node to delete.
freeItem	if !0 then item is freed using free function. If 0 (or free function is NULL) then item is not freed.

References LINKEDLIST::free_func, LINKEDLIST::head, LINKEDLIST::size, and LINKEDLIST::tail.

9.156.2.6 ListDestroy()

```
int ListDestroy (
```

```
            LinkedList * list,
            int freeItem )
```

Removes all memory associated with list nodes. Does not free LinkedList *list.

Precondition: The list has been initialized.

Returns

0 on success, EINVAL on failure.

Parameters

list	Must be valid, non null, pointer to a linked list.
freeItem	if !0 then item is freed using free function. If 0 (or free function is NULL) then item is not freed.

References LINKEDLIST::freeNodeList, LINKEDLIST::head, LINKEDLIST::size, and LINKEDLIST::tail.

9.156.2.7 ListFind()

```
ListNode * ListFind (
            LinkedList * list,
            ListNode * start,
            void * item )
```

Finds the specified item in the list.

Uses the compare function specified in ListInit. If compare function is NULL then compares items as pointers.

Precondition: The list has been initialized.

Returns

The node containing the item. NULL if no node contains the item.

Parameters

list	Must be valid, non null, pointer to a linked list.
start	The node to start from, NULL if to start from beginning.
item	The item to search for.

References LINKEDLIST::cmp_func, LINKEDLIST::head, and LINKEDLIST::tail.

9.156.2.8 ListHead()

```
ListNode * ListHead (
            LinkedList * list )
```

Returns the head of the list.

Precondition: The list has been initialized.

Returns

The head of the list. NULL if list is empty.

Parameters

list	Must be valid, non null, pointer to a linked list.

References LINKEDLIST::head, and LINKEDLIST::size.

9.156.2.9 ListInit()

```
int ListInit (
            LinkedList * list,
            cmp_routine cmp_func,
            free_function free_func )
```

Initializes LinkedList. Must be called first and only once for List.

Returns

- 0 on success.
- EOUTOFMEM on failure.

Parameters

list	Must be valid, non null, pointer to a linked list.
cmp_func	Function used to compare items. (May be NULL).
free_func	Function used to free items. (May be NULL).

References LINKEDLIST::cmp_func, LINKEDLIST::free_func, LINKEDLIST::freeNodeList, LINKEDLIST::head, LINKEDLIST::size, and LINKEDLIST::tail.

9.156.2.10 ListNext()

```
ListNode * ListNext (
            LinkedList * list,
            ListNode * node )
```

Returns the next item in the list.

Precondition: The list has been initialized.

Returns

The next item in the list. NULL if there are no more items in list.

Parameters

list	Must be valid, non null, pointer to a linked list.
node	Node from the list.

References LINKEDLIST::tail.

9.156.2.11 ListPrev()

```
ListNode * ListPrev (
            LinkedList * list,
            ListNode * node )
```

Returns the previous item in the list.

Precondition: The list has been initialized.

Returns

The previous item in the list. NULL if there are no more items in list.

Parameters

list	Must be valid, non null, pointer to a linked list.
node	Node from the list.

References LINKEDLIST::head.

9.156.2.12 ListSize()

```
long ListSize (
            LinkedList * list )
```

Returns the size of the list.

Precondition: The list has been initialized.

Returns

The number of items in the list.

Parameters

list	Must be valid, non null, pointer to a linked list.

References LINKEDLIST::size.

9.156.2.13 ListTail()

```
ListNode * ListTail (
            LinkedList * list )
```

Returns the tail of the list.

Precondition: The list has been initialized.

Returns

The tail of the list. NULL if list is empty.

Parameters

list	Must be valid, non null, pointer to a linked list.

References LINKEDLIST::size, and LINKEDLIST::tail.

9.157 LinkedList.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * * Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * * Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * * Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  *****************************************************************************/
00031
00032 #ifndef LINKED_LIST_H
00033 #define LINKED_LIST_H
00034
00039 #include "FreeList.h"
00040
00041 #ifdef __cplusplus
00042 extern "C" {
00043 #endif
00044
00045 #define EOUTOFMEM (-7 & 1 « 29)
00046
00047 #define FREELISTSIZE 100
00048 #define LIST_SUCCESS 1
00049 #define LIST_FAIL 0
00050
00052 typedef void (*free_function)(void *arg);
00053
00055 typedef int (*cmp_routine)(void *itemA, void *itemB);
00056
00060 typedef struct LISTNODE
00061 {
00062         struct LISTNODE *prev;
00063         struct LISTNODE *next;
00064         void *item;
00065 } ListNode;
00066
00083 typedef struct LINKEDLIST
00084 {
00086         ListNode head;
00088         ListNode tail;
00090         long size;
```

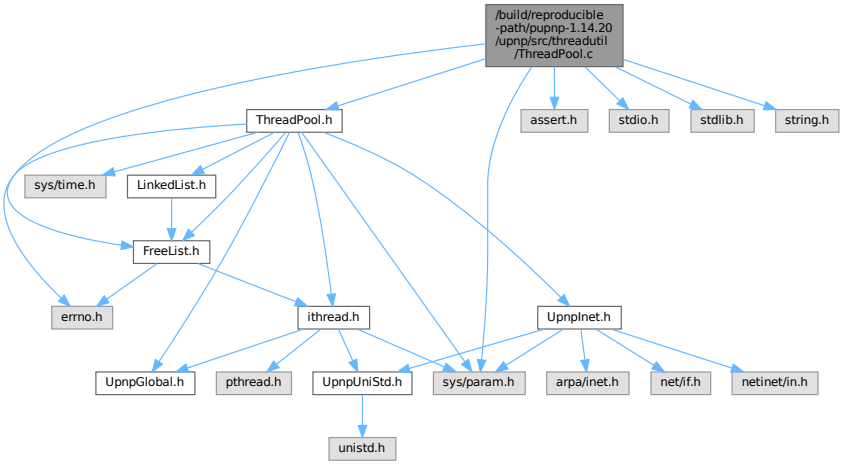
```
00092         FreeList freeNodeList;
00094         free_function free_func;
00096         cmp_routine cmp_func;
00097 } LinkedList;
00098
00106 int ListInit(
00108         LinkedList *list,
00110         cmp_routine cmp_func,
00112         free_function free_func);
00113
00123 ListNode *ListAddHead(
00125         LinkedList *list,
00127         void *item);
00128
00137 ListNode *ListAddTail(
00139         LinkedList *list,
00141         void *item);
00142
00151 ListNode *ListAddAfter(
00153         LinkedList *list,
00155         void *item,
00157         ListNode *bnode);
00158
00167 ListNode *ListAddBefore(
00169         LinkedList *list,
00171         void *item,
00173         ListNode *anode);
00174
00183 void *ListDelNode(
00185         LinkedList *list,
00187         ListNode *dnode,
00190         int freeItem);
00191
00200 int ListDestroy(
00202         LinkedList *list,
00205         int freeItem);
00206
00214 ListNode *ListHead(
00216         LinkedList *list);
00217
00225 ListNode *ListTail(
00227         LinkedList *list);
00228
00236 ListNode *ListNext(
00238         LinkedList *list,
00240         ListNode *node);
00241
00250 ListNode *ListPrev(
00252         LinkedList *list,
00254         ListNode *node);
00255
00266 ListNode *ListFind(
00268         LinkedList *list,
00270         ListNode *start,
00272         void *item);
00273
00281 long ListSize(
00283         LinkedList *list);
00284
00285 #ifdef __cplusplus
00286 }
00287 #endif
00288
00289 #endif /* LINKED_LIST_H */
```

9.158 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/↩ ThreadPool.c File Reference

```
#include <sys/param.h>
#include "ThreadPool.h"
#include "FreeList.h"
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
```

Include dependency graph for ThreadPool.c:



Functions

- static long DiffMillis (struct timeval *time1, struct timeval *time2)

  Returns the difference in milliseconds between two timeval structures.
- static void StatsInit (ThreadPoolStats *stats)

  Initializes the statistics structure.
- static void StatsAccountLQ (ThreadPool *tp, long diffTime)
- static void StatsAccountMQ (ThreadPool *tp, long diffTime)
- static void StatsAccountHQ (ThreadPool *tp, long diffTime)
- static void CalcWaitTime (ThreadPool *tp, ThreadPriority p, ThreadPoolJob *job)

  Calculates the time the job has been waiting at the specified priority.
- static time_t StatsTime (time_t *t)
- static int CmpThreadPoolJob (void *jobA, void *jobB)

  Compares thread pool jobs.
- static void FreeThreadPoolJob (ThreadPool *tp, ThreadPoolJob *tpj)

  Deallocates a dynamically allocated ThreadPoolJob.
- static int SetPolicyType (PolicyType in)

  Sets the scheduling policy of the current process.
- static int SetPriority (ThreadPriority priority)

  Sets the priority of the currently running thread.
- static void BumpPriority (ThreadPool *tp)

  Determines whether any jobs need to be bumped to a higher priority Q and bumps them.
- static void SetRelTimeout (struct timespec *time, int relMillis)

  Sets the fields of the passed in timespec to be relMillis milliseconds in the future.
- static void SetSeed (void)

  Sets seed for random number generator. Each thread sets the seed random number generator.
- static void * WorkerThread (void *arg)

  Implements a thread pool worker. Worker waits for a job to become available. Worker picks up persistent jobs first, high priority, med priority, then low priority.
- static ThreadPoolJob * CreateThreadPoolJob (ThreadPoolJob *job, int id, ThreadPool *tp)

  Creates a Thread Pool Job. (Dynamically allocated)

- static int CreateWorker (ThreadPool *tp)

  Creates a worker thread, if the thread pool does not already have max threads.
- static void AddWorker (ThreadPool *tp)

  Determines whether or not a thread should be added based on the jobsPerThread ratio. Adds a thread if appropriate.
- int ThreadPoolInit (ThreadPool *tp, ThreadPoolAttr *attr)

  Initializes and starts ThreadPool. Must be called first and only once for ThreadPool.
- int ThreadPoolAddPersistent (ThreadPool *tp, ThreadPoolJob *job, int *jobId)

  Adds a persistent job to the thread pool.
- int ThreadPoolAdd (ThreadPool *tp, ThreadPoolJob *job, int *jobId)

  Adds a job to the thread pool. Job will be run as soon as possible.
- int ThreadPoolRemove (ThreadPool *tp, int jobId, ThreadPoolJob *out)

  Removes a job from the thread pool. Can only remove jobs which are not currently running.
- int ThreadPoolGetAttr (ThreadPool *tp, ThreadPoolAttr *out)

  Gets the current set of attributes associated with the thread pool.
- int ThreadPoolSetAttr (ThreadPool *tp, ThreadPoolAttr *attr)

  Sets the attributes for the thread pool. Only affects future calculations.
- int ThreadPoolShutdown (ThreadPool *tp)

  Shuts the thread pool down. Waits for all threads to finish. May block indefinitely if jobs do not exit.
- void TPSetMaxJobsTotal (int mjt)

  Sets the maximum number of jobs in the thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.
- int TPAttrInit (ThreadPoolAttr *attr)

  Initializes thread pool attributes. Sets values to defaults defined in ThreadPool.h.
- int TPJobInit (ThreadPoolJob *job, start_routine func, void *arg)

  Initializes thread pool job. Sets the priority to default defined in ThreadPool.h. Sets the free_routine to default defined in ThreadPool.h.
- int TPJobSetPriority (ThreadPoolJob *job, ThreadPriority priority)

  Sets the max threads for the thread pool attributes.
- int TPJobSetFreeFunction (ThreadPoolJob *job, free_routine func)

  Sets the max threads for the thread pool attributes.
- int TPAttrSetMaxThreads (ThreadPoolAttr *attr, int maxThreads)

  Sets the max threads for the thread pool attributes.
- int TPAttrSetMinThreads (ThreadPoolAttr *attr, int minThreads)

  Sets the min threads for the thread pool attributes.
- int TPAttrSetStackSize (ThreadPoolAttr *attr, size_t stackSize)

  Sets the stack size for the thread pool attributes.
- int TPAttrSetIdleTime (ThreadPoolAttr *attr, int idleTime)

  Sets the idle time for the thread pool attributes.
- int TPAttrSetJobsPerThread (ThreadPoolAttr *attr, int jobsPerThread)

  Sets the jobs per thread ratio.
- int TPAttrSetStarvationTime (ThreadPoolAttr *attr, int starvationTime)

  Sets the starvation time for the thread pool attributes.
- int TPAttrSetSchedPolicy (ThreadPoolAttr *attr, PolicyType schedPolicy)

  Sets the scheduling policy for the thread pool attributes.
- int TPAttrSetMaxJobsTotal (ThreadPoolAttr *attr, int maxJobsTotal)

  Sets the maximum number jobs that can be qeued totally.
- void ThreadPoolPrintStats (ThreadPoolStats *stats)
- int ThreadPoolGetStats (ThreadPool *tp, ThreadPoolStats *stats)

  Returns various statistics about the thread pool.

Variables

- int **maxJobsTotal** = 100

9.158.1 Function Documentation

9.158.1.1 AddWorker()

```
static void AddWorker (
            ThreadPool * tp )  [static]
```

Determines whether or not a thread should be added based on the jobsPerThread ratio. Adds a thread if appropriate.

Remarks

The ThreadPool object mutex must be locked prior to calling this function.

Parameters

tp	A pointer to the ThreadPool object.

References THREADPOOL::attr, THREADPOOL::busyThreads, CreateWorker(), THREADPOOL::highJobQ, THREADPOOLATTR::jobsPerThread, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOL::persistentThreads, LINKEDLIST::size, and THREADPOOL::totalThreads.

Referenced by ThreadPoolAdd().

9.158.1.2 BumpPriority()

```
static void BumpPriority (
            ThreadPool * tp )  [static]
```

Determines whether any jobs need to be bumped to a higher priority Q and bumps them.

tp->mutex must be locked.

Returns

Parameters

tp	.

References THREADPOOL::attr, DiffMillis(), LINKEDLIST::head, THREADPOOL::highJobQ, THREADPOOL::lowJobQ, THREADPOOLATTR::maxIdleTime, THREADPOOL::medJobQ, LINKEDLIST::size, THREADPOOLATTR::starvationTime, StatsAccountLQ(), and StatsAccountMQ().

Referenced by WorkerThread().

9.158.1.3 CalcWaitTime()

```
static void CalcWaitTime (
             ThreadPool * tp,
             ThreadPriority p,
             ThreadPoolJob * job )  [static]
```

Calculates the time the job has been waiting at the specified priority.

Adds to the totalTime and totalJobs kept in the thread pool statistics structure.

Parameters

tp	.
p	.
job	.

References DiffMillis(), StatsAccountHQ(), StatsAccountLQ(), and StatsAccountMQ().

Referenced by WorkerThread().

9.158.1.4 CmpThreadPoolJob()

```
static int CmpThreadPoolJob (
             void * jobA,
             void * jobB )  [static]
```

Compares thread pool jobs.

Referenced by ThreadPoolInit().

9.158.1.5 CreateThreadPoolJob()

```
static ThreadPoolJob * CreateThreadPoolJob (
             ThreadPoolJob * job,
             int id,
             ThreadPool * tp )  [static]
```

Creates a Thread Pool Job. (Dynamically allocated)

Returns

ThreadPoolJob *on success, NULL on failure.

Parameters

job	job is copied.
id	id of job.
tp	.

References THREADPOOL::jobFreeList.

Referenced by ThreadPoolAdd(), and ThreadPoolAddPersistent().

9.158.1.6 CreateWorker()

```
static int CreateWorker (
            ThreadPool * tp )  [static]
```

Creates a worker thread, if the thread pool does not already have max threads.

Remarks

The ThreadPool object mutex must be locked prior to calling this function.

Returns

- 0 on success, < 0 on failure.
- EMAXTHREADS if already max threads reached.
- EAGAIN if system can not create thread.

Parameters

tp	A pointer to the ThreadPool object.

References THREADPOOL::attr, THREADPOOLATTR::maxThreads, THREADPOOL::mutex, THREADPOOL::pendingWorkerThread
THREADPOOLATTR::stackSize, THREADPOOL::start_and_shutdown, THREADPOOL::stats, THREADPOOL::totalThreads,
and WorkerThread().

Referenced by AddWorker(), ThreadPoolAddPersistent(), ThreadPoolInit(), and ThreadPoolSetAttr().

9.158.1.7 DiffMillis()

```
static long DiffMillis (
            struct timeval * time1,
            struct timeval * time2 )  [static]
```

Returns the difference in milliseconds between two timeval structures.

Returns

The difference in milliseconds, time1-time2.

Parameters

time1	.
time2	.

Referenced by BumpPriority(), and CalcWaitTime().

9.158.1.8 FreeThreadPoolJob()

```
static void FreeThreadPoolJob (
             ThreadPool * tp,
             ThreadPoolJob * tpj )  [static]
```

Deallocates a dynamically allocated ThreadPoolJob.

Parameters

tp	.
tpj	Must be allocated with CreateThreadPoolJob.

References THREADPOOL::jobFreeList.

Referenced by ThreadPoolAdd(), ThreadPoolRemove(), ThreadPoolShutdown(), and WorkerThread().

9.158.1.9 SetPolicyType()

```
static int SetPolicyType (
             PolicyType in )  [static]
```

Sets the scheduling policy of the current process.

Returns

- `0` on success.
- `result` of GetLastError() on failure.

Parameters

in	.

Referenced by ThreadPoolInit(), and ThreadPoolSetAttr().

9.158.1.10 SetPriority()

```
static int SetPriority (
             ThreadPriority priority )  [static]
```

Sets the priority of the currently running thread.

Returns

- `0` on success.
- `EINVAL` invalid priority or the result of GerLastError.

Parameters

priority	.

Referenced by WorkerThread().

9.158.1.11 SetRelTimeout()

```
static void SetRelTimeout (
            struct timespec * time,
            int relMillis )  [static]
```

Sets the fields of the passed in timespec to be relMillis milliseconds in the future.

Parameters

time	.
relMillis	milliseconds in the future.

Referenced by WorkerThread().

9.158.1.12 SetSeed()

```
static void SetSeed (
            void  )  [static]
```

Sets seed for random number generator. Each thread sets the seed random number generator.

Referenced by WorkerThread().

9.158.1.13 StatsAccountHQ()

```
static void StatsAccountHQ (
            ThreadPool * tp,
            long diffTime )  [static]
```

Parameters

tp	.
diffTime	.

References THREADPOOL::stats.

Referenced by CalcWaitTime().

9.158.1.14 StatsAccountLQ()

```
static void StatsAccountLQ (
```

```
            ThreadPool * tp,
            long diffTime )  [static]
```

Parameters

tp	.
diffTime	.

References THREADPOOL::stats.

Referenced by BumpPriority(), and CalcWaitTime().

9.158.1.15 StatsAccountMQ()

```
static void StatsAccountMQ (
            ThreadPool * tp,
            long diffTime )  [static]
```

Parameters

tp	.
diffTime	.

References THREADPOOL::stats.

Referenced by BumpPriority(), and CalcWaitTime().

9.158.1.16 StatsInit()

```
static void StatsInit (
            ThreadPoolStats * stats )  [static]
```

Initializes the statistics structure.

Parameters

stats	Must be valid non null stats structure.

Referenced by ThreadPoolInit().

9.158.1.17 StatsTime()

```
static time_t StatsTime (
            time_t * t )  [static]
```

Parameters

t	.

Referenced by ThreadPoolPrintStats(), and WorkerThread().

9.158.1.18 ThreadPoolAdd()

```
int ThreadPoolAdd (
             ThreadPool * tp,
             ThreadPoolJob * job,
             int * jobId )
```

Adds a job to the thread pool. Job will be run as soon as possible.

Returns

- 0 on success, nonzero on failure.
- EOUTOFMEM if not enough memory to add job.

Parameters

tp	valid thread pool pointer.
job	.
jobId	id of job.

References AddWorker(), THREADPOOL::attr, THREADPOOL::condition, CreateThreadPoolJob(), FreeThreadPoolJob(), THREADPOOL::highJobQ, INVALID_JOB_ID, THREADPOOL::lastJobId, THREADPOOL::lowJobQ, THREADPOOLATTR::maxJobs THREADPOOL::medJobQ, THREADPOOL::mutex, and LINKEDLIST::size.

Referenced by genaNotifyThread(), readFromSSDPSocket(), ssdp_handle_ctrlpt_msg(), TimerThreadWorker(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.158.1.19 ThreadPoolAddPersistent()

```
int ThreadPoolAddPersistent (
             ThreadPool * tp,
             ThreadPoolJob * job,
             int * jobId )
```

Adds a persistent job to the thread pool.

Job will be run as soon as possible. Call will block until job is scheduled.

Returns

- 0 on success.
- EOUTOFMEM not enough memory to add job.
- EMAXTHREADS not enough threads to add persistent job.

Parameters

tp	Valid thread pool pointer.
job	Valid thread pool job.
job_id	.

References THREADPOOL::attr, THREADPOOL::condition, CreateThreadPoolJob(), CreateWorker(), INVALID_JOB_ID, THREADPOOL::lastJobId, THREADPOOLATTR::maxThreads, THREADPOOL::mutex, THREADPOOL::persistentJob, THREADPOOL::persistentThreads, THREADPOOL::start_and_shutdown, and THREADPOOL::totalThreads.

Referenced by StartMiniServer(), TimerThreadInit(), and TimerThreadWorker().

9.158.1.20 ThreadPoolGetAttr()

```
int ThreadPoolGetAttr (
            ThreadPool * tp,
            ThreadPoolAttr * out )
```

Gets the current set of attributes associated with the thread pool.

Returns

- 0 on success, nonzero on failure.

Parameters

tp	valid thread pool pointer.
out	non null pointer to store attributes.

References THREADPOOL::attr, THREADPOOL::mutex, and THREADPOOL::shutdown.

9.158.1.21 ThreadPoolGetStats()

```
int ThreadPoolGetStats (
            ThreadPool * tp,
            ThreadPoolStats * stats )
```

Returns various statistics about the thread pool.

Only valid if STATS has been defined.

Returns

Always returns 0.

Parameters

tp	Valid initialized threadpool.
stats	Valid stats, out parameter.

References THREADPOOL::highJobQ, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOL::mutex, THREADPOOL::persistentThreads, THREADPOOL::shutdown, THREADPOOL::stats, and THREADPOOL::totalThreads.

Referenced by PrintThreadPoolStats().

9.158.1.22 ThreadPoolInit()

```
int ThreadPoolInit (
            ThreadPool * tp,
            ThreadPoolAttr * attr )
```

Initializes and starts ThreadPool. Must be called first and only once for ThreadPool.

Returns

- 0 on success.
- EAGAIN if not enough system resources to create minimum threads.
- INVALID_POLICY if schedPolicy can't be set.
- EMAXTHREADS if minimum threads is greater than maximum threads.

Parameters

tp	Must be valid, non null, pointer to ThreadPool.
attr	Can be null. if not null then attr contains the following fields: • minWorkerThreads - minimum number of worker threads thread pool will never have less than this number of threads. • maxWorkerThreads - maximum number of worker threads thread pool will never have more than this number of threads. • maxIdleTime - maximum time that a worker thread will spend idle. If a worker is idle longer than this time and there are more than the min number of workers running, then the worker thread exits. • jobsPerThread - ratio of jobs to thread to try and maintain if a job is scheduled and the number of jobs per thread is greater than this number,and if less than the maximum number of workers are running then a new thread is started to help out with efficiency. • schedPolicy - scheduling policy to try and set (OS dependent).

References THREADPOOL::attr, THREADPOOL::busyThreads, CmpThreadPoolJob(), THREADPOOL::condition, CreateWorker(), THREADPOOL::highJobQ, INVALID_POLICY, THREADPOOL::jobFreeList, JOBFREELISTSIZE, THREADPOOL::lastJobId, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOLATTR::minThreads, THREADPOOL::mutex, THREADPOOL::pendingWorkerThreadStart, THREADPOOL::persistentJob, THREADPOOL::persistentThrea THREADPOOLATTR::schedPolicy, SetPolicyType(), THREADPOOL::shutdown, THREADPOOL::start_and_shutdown, THREADPOOL::stats, StatsInit(), ThreadPoolShutdown(), THREADPOOL::totalThreads, and TPAttrInit().

Referenced by UpnpInitThreadPools().

9.158.1.23 ThreadPoolPrintStats()

```
void ThreadPoolPrintStats (
            ThreadPoolStats * stats )
```

Parameters

stats	.

References StatsTime().

9.158.1.24 ThreadPoolRemove()

```
int ThreadPoolRemove (
            ThreadPool * tp,
            int jobId,
            ThreadPoolJob * out )
```

Removes a job from the thread pool. Can only remove jobs which are not currently running.

Returns

- 0 on success, nonzero on failure.
- INVALID_JOB_ID if job not found.

Parameters

tp	valid thread pool pointer.
jobId	id of job.
out	space for removed job.

References FreeThreadPoolJob(), THREADPOOL::highJobQ, INVALID_JOB_ID, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOL::mutex, and THREADPOOL::persistentJob.

9.158.1.25 ThreadPoolSetAttr()

```
int ThreadPoolSetAttr (
            ThreadPool * tp,
            ThreadPoolAttr * attr )
```

Sets the attributes for the thread pool. Only affects future calculations.

Returns

- 0 on success, nonzero on failure.
- INVALID_POLICY if policy can not be set.

Parameters

tp	valid thread pool pointer.
attr	pointer to attributes, null sets attributes to default.

References THREADPOOL::attr, THREADPOOL::condition, CreateWorker(), INVALID_POLICY, THREADPOOLATTR::minThreads, THREADPOOL::mutex, THREADPOOLATTR::schedPolicy, SetPolicyType(), ThreadPoolShutdown(), THREADPOOL::totalThreads, and TPAttrInit().

9.158.1.26 ThreadPoolShutdown()

```
int ThreadPoolShutdown (
            ThreadPool * tp )
```

Shuts the thread pool down. Waits for all threads to finish. May block indefinitely if jobs do not exit.

Returns

0 on success, nonzero on failure

Parameters

tp	must be valid tp.

References THREADPOOL::condition, FreeThreadPoolJob(), THREADPOOL::highJobQ, THREADPOOL::jobFreeList, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOL::mutex, THREADPOOL::persistentJob, THREADPOOL::shutdown, LINKEDLIST::size, THREADPOOL::start_and_shutdown, and THREADPOOL::totalThreads.

Referenced by ThreadPoolInit(), ThreadPoolSetAttr(), and UpnpFinish().

9.158.1.27 TPAttrInit()

```
int TPAttrInit (
            ThreadPoolAttr * attr )
```

Initializes thread pool attributes. Sets values to defaults defined in ThreadPool.h.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.

References DEFAULT_IDLE_TIME, DEFAULT_JOBS_PER_THREAD, DEFAULT_MAX_THREADS, DEFAULT_MIN_THREADS, DEFAULT_STACK_SIZE, DEFAULT_STARVATION_TIME, THREADPOOLATTR::jobsPerThread, THREADPOOLATTR::maxIdleTime, THREADPOOLATTR::maxJobsTotal, THREADPOOLATTR::maxThreads, THREADPOOLATTR::minThreads, THREADPOOLATTR::schedPolicy, THREADPOOLATTR::stackSize, and THREADPOOLATTR::starvationTime.

Referenced by ThreadPoolInit(), ThreadPoolSetAttr(), and UpnpInitThreadPools().

9.158.1.28 TPAttrSetIdleTime()

```
int TPAttrSetIdleTime (
```

```
            ThreadPoolAttr * attr,
            int idleTime )
```

Sets the idle time for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
idleTime	.

References THREADPOOLATTR::maxIdleTime.

Referenced by UpnpInitThreadPools().

9.158.1.29 TPAttrSetJobsPerThread()

```
int TPAttrSetJobsPerThread (
            ThreadPoolAttr * attr,
            int jobsPerThread )
```

Sets the jobs per thread ratio.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
jobsPerThread	number of jobs per thread to maintain.

References THREADPOOLATTR::jobsPerThread.

Referenced by UpnpInitThreadPools().

9.158.1.30 TPAttrSetMaxJobsTotal()

```
int TPAttrSetMaxJobsTotal (
            ThreadPoolAttr * attr,
            int maxJobsTotal )
```

Sets the maximum number jobs that can be qeued totally.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
maxJobsTotal	maximum number of jobs.

References THREADPOOLATTR::maxJobsTotal.

Referenced by UpnpInitThreadPools().

9.158.1.31 TPAttrSetMaxThreads()

```
int TPAttrSetMaxThreads (
            ThreadPoolAttr * attr,
            int maxThreads )
```

Sets the max threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
maxThreads	value to set.

References THREADPOOLATTR::maxThreads.

Referenced by UpnpInitThreadPools().

9.158.1.32 TPAttrSetMinThreads()

```
int TPAttrSetMinThreads (
            ThreadPoolAttr * attr,
            int minThreads )
```

Sets the min threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
minThreads	value to set.

References THREADPOOLATTR::minThreads.

Referenced by UpnpInitThreadPools().

9.158.1.33 TPAttrSetSchedPolicy()

```
int TPAttrSetSchedPolicy (
            ThreadPoolAttr * attr,
            PolicyType schedPolicy )
```

Sets the scheduling policy for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
schedPolicy	must be a valid policy type.

References THREADPOOLATTR::schedPolicy.

9.158.1.34 TPAttrSetStackSize()

```
int TPAttrSetStackSize (
            ThreadPoolAttr * attr,
            size_t stackSize )
```

Sets the stack size for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
stackSize	value to set.

References THREADPOOLATTR::stackSize.

Referenced by UpnpInitThreadPools().

9.158.1.35 TPAttrSetStarvationTime()

```
int TPAttrSetStarvationTime (
            ThreadPoolAttr * attr,
            int starvationTime )
```

Sets the starvation time for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
starvationTime	milliseconds.

References THREADPOOLATTR::starvationTime.

9.158.1.36 TPJobInit()

```
int TPJobInit (
            ThreadPoolJob * job,
            start_routine func,
            void * arg )
```

Initializes thread pool job. Sets the priority to default defined in ThreadPool.h. Sets the free_routine to default defined in ThreadPool.h.

Returns

Always returns 0.

Parameters

job	must be valid thread pool attributes.
func	function to run, must be valid.
arg	argument to pass to function.

References DEFAULT_FREE_ROUTINE, and DEFAULT_PRIORITY.

Referenced by readFromSSDPSocket(), ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_ctrlpt_msg(), ssdp_handle_device_request(), StartMiniServer(), TimerThreadInit(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsy UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.158.1.37 TPJobSetFreeFunction()

```
int TPJobSetFreeFunction (
            ThreadPoolJob * job,
            free_routine func )
```

Sets the max threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

job	must be valid thread pool attributes.
func	value to set.

Referenced by readFromSSDPSocket(), ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_ctrlpt_msg(), ssdp_handle_device_request(), StartMiniServer(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.158.1.38 TPJobSetPriority()

```
int TPJobSetPriority (
            ThreadPoolJob * job,
            ThreadPriority priority )
```

Sets the max threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

job	must be valid thread pool attributes.
priority	value to set.

Referenced by readFromSSDPSocket(), ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_ctrlpt_msg(), StartMiniServer(), TimerThreadInit(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.158.1.39 TPSetMaxJobsTotal()

```
void TPSetMaxJobsTotal (
            int mjt )
```

Sets the maximum number of jobs in the thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.

Referenced by UpnpSetMaxJobsTotal().

9.158.1.40 WorkerThread()

```
static void * WorkerThread (
            void * arg )  [static]
```

Implements a thread pool worker. Worker waits for a job to become available. Worker picks up persistent jobs first, high priority, med priority, then low priority.

If worker remains idle for more than specified max, the worker is released.

Parameters

arg	arg -> is cast to (ThreadPool *).

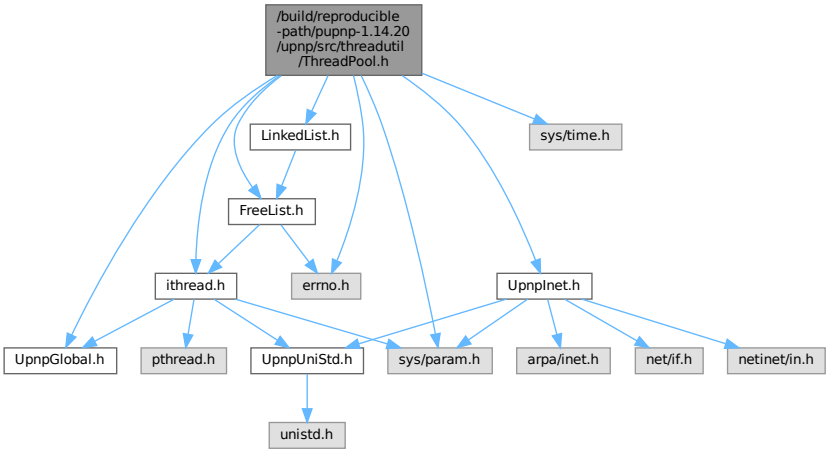
References THREADPOOL::attr, BumpPriority(), THREADPOOL::busyThreads, CalcWaitTime(), THREADPOOL::condition, DEFAULT_PRIORITY, FreeThreadPoolJob(), THREADPOOL::highJobQ, THREADPOOL::lowJobQ, THREADPOOLATTR::maxIdleTi THREADPOOLATTR::maxThreads, THREADPOOL::medJobQ, THREADPOOLATTR::minThreads, THREADPOOL::mutex, THREADPOOL::pendingWorkerThreadStart, THREADPOOL::persistentJob, THREADPOOL::persistentThreads, SetPriority(), SetRelTimeout(), SetSeed(), THREADPOOL::shutdown, LINKEDLIST::size, THREADPOOL::start_and_shutdown, THREADPOOL::stats, StatsTime(), and THREADPOOL::totalThreads.

Referenced by CreateWorker().

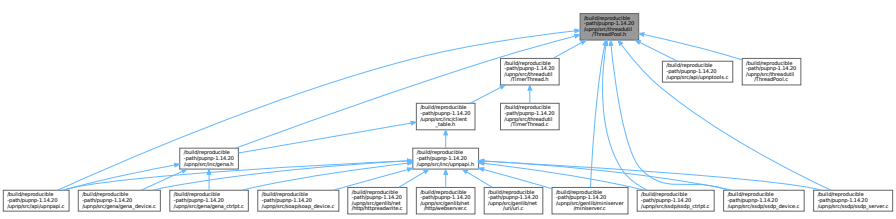
9.159 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/↩ ThreadPool.h File Reference

```
#include "FreeList.h"
#include "LinkedList.h"
#include "UpnpGlobal.h"
#include "UpnpInet.h"
#include "ithread.h"
#include <errno.h>
#include <sys/param.h>
#include <sys/time.h>
```

Include dependency graph for ThreadPool.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct THREADPOOLATTR
- struct THREADPOOLJOB
- struct TPOOLSTATS
- struct THREADPOOL

  A thread pool similar to the thread pool in the UPnP SDK.

Macros

- #define JOBFREELISTSIZE 100
- #define **INFINITE_THREADS** -1
- #define **EMAXTHREADS** (-8 & 1 << 29)
- #define INVALID_POLICY (-9 & 1 << 29)
- #define INVALID_JOB_ID (-2 & 1 << 29)
- #define DEFAULT_PRIORITY MED_PRIORITY
- #define DEFAULT_MIN_THREADS 1
- #define DEFAULT_MAX_THREADS 10
- #define DEFAULT_STACK_SIZE 0u
- #define DEFAULT_JOBS_PER_THREAD 10
- #define DEFAULT_STARVATION_TIME 500
- #define DEFAULT_IDLE_TIME 10 * 1000
- #define DEFAULT_FREE_ROUTINE NULL
- #define DEFAULT_MAX_JOBS_TOTAL 100
- #define STATS 1

  Statistics.
- #define **DEFAULT_POLICY** SCHED_OTHER

Typedefs

- typedef enum duration **Duration**
- typedef enum priority **ThreadPriority**
- typedef int **PolicyType**
- typedef void(* free_routine) (void *arg)
- typedef struct THREADPOOLATTR ThreadPoolAttr
- typedef struct THREADPOOLJOB ThreadPoolJob
- typedef struct TPOOLSTATS ThreadPoolStats
- typedef struct THREADPOOL ThreadPool

  A thread pool similar to the thread pool in the UPnP SDK.

Enumerations

- enum **duration** { **SHORT_TERM** , **PERSISTENT** }
- enum **priority** { **LOW_PRIORITY** , **MED_PRIORITY** , **HIGH_PRIORITY** }

Functions

- void TPSetMaxJobsTotal (int mjt)

  Sets the maximum number of jobs in the thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.
- int ThreadPoolInit (ThreadPool *tp, ThreadPoolAttr *attr)

  Initializes and starts ThreadPool. Must be called first and only once for ThreadPool.
- int ThreadPoolAddPersistent (ThreadPool *tp, ThreadPoolJob *job, int *jobId)

  Adds a persistent job to the thread pool.
- int ThreadPoolGetAttr (ThreadPool *tp, ThreadPoolAttr *out)

  Gets the current set of attributes associated with the thread pool.
- int ThreadPoolSetAttr (ThreadPool *tp, ThreadPoolAttr *attr)

  Sets the attributes for the thread pool. Only affects future calculations.
- int ThreadPoolAdd (ThreadPool *tp, ThreadPoolJob *job, int *jobId)

  Adds a job to the thread pool. Job will be run as soon as possible.
- int ThreadPoolRemove (ThreadPool *tp, int jobId, ThreadPoolJob *out)

  Removes a job from the thread pool. Can only remove jobs which are not currently running.
- int ThreadPoolShutdown (ThreadPool *tp)

  Shuts the thread pool down. Waits for all threads to finish. May block indefinitely if jobs do not exit.
- int TPJobInit (ThreadPoolJob *job, start_routine func, void *arg)

  Initializes thread pool job. Sets the priority to default defined in ThreadPool.h. Sets the free_routine to default defined in ThreadPool.h.
- int TPJobSetPriority (ThreadPoolJob *job, ThreadPriority priority)

  Sets the max threads for the thread pool attributes.
- int TPJobSetFreeFunction (ThreadPoolJob *job, free_routine func)

  Sets the max threads for the thread pool attributes.
- int TPAttrInit (ThreadPoolAttr *attr)

  Initializes thread pool attributes. Sets values to defaults defined in ThreadPool.h.
- int TPAttrSetMaxThreads (ThreadPoolAttr *attr, int maxThreads)

  Sets the max threads for the thread pool attributes.
- int TPAttrSetMinThreads (ThreadPoolAttr *attr, int minThreads)

  Sets the min threads for the thread pool attributes.
- int TPAttrSetStackSize (ThreadPoolAttr *attr, size_t stackSize)

  Sets the stack size for the thread pool attributes.
- int TPAttrSetIdleTime (ThreadPoolAttr *attr, int idleTime)

  Sets the idle time for the thread pool attributes.
- int TPAttrSetJobsPerThread (ThreadPoolAttr *attr, int jobsPerThread)

  Sets the jobs per thread ratio.
- int TPAttrSetStarvationTime (ThreadPoolAttr *attr, int starvationTime)

  Sets the starvation time for the thread pool attributes.
- int TPAttrSetSchedPolicy (ThreadPoolAttr *attr, PolicyType schedPolicy)

  Sets the scheduling policy for the thread pool attributes.
- int TPAttrSetMaxJobsTotal (ThreadPoolAttr *attr, int maxJobsTotal)

  Sets the maximum number jobs that can be qeued totally.
- UPNP_EXPORT_SPEC int ThreadPoolGetStats (ThreadPool *tp, ThreadPoolStats *stats)

  Returns various statistics about the thread pool.
- UPNP_EXPORT_SPEC void ThreadPoolPrintStats (ThreadPoolStats *stats)

Variables

- int **maxJobsTotal**

9.159.1 Macro Definition Documentation

9.159.1.1 DEFAULT_FREE_ROUTINE

```
#define DEFAULT_FREE_ROUTINE NULL
```

default free routine used TPJobInit

9.159.1.2 DEFAULT_IDLE_TIME

```
#define DEFAULT_IDLE_TIME 10 * 1000
```

default idle time used by TPAttrInit

9.159.1.3 DEFAULT_JOBS_PER_THREAD

```
#define DEFAULT_JOBS_PER_THREAD 10
```

default jobs per thread used by TPAttrInit

9.159.1.4 DEFAULT_MAX_JOBS_TOTAL

```
#define DEFAULT_MAX_JOBS_TOTAL 100
```

default max jobs used TPAttrInit

9.159.1.5 DEFAULT_MAX_THREADS

```
#define DEFAULT_MAX_THREADS 10
```

default max used by TPAttrInit

9.159.1.6 DEFAULT_MIN_THREADS

```
#define DEFAULT_MIN_THREADS 1
```

default minimum used by TPAttrInit

9.159.1.7 DEFAULT_PRIORITY

```
#define DEFAULT_PRIORITY MED_PRIORITY
```

default priority used by TPJobInit

9.159.1.8 DEFAULT_STACK_SIZE

```
#define DEFAULT_STACK_SIZE 0u
```

default stack size used by TPAttrInit

9.159.1.9 DEFAULT_STARVATION_TIME

```
#define DEFAULT_STARVATION_TIME 500
```

default starvation time used by TPAttrInit

9.159.1.10 INVALID_JOB_ID

```
#define INVALID_JOB_ID (-2 & 1 << 29)
```

Invalid JOB Id

9.159.1.11 INVALID_POLICY

```
#define INVALID_POLICY (-9 & 1 << 29)
```

Invalid Policy

9.159.1.12 JOBFREELISTSIZE

```
#define JOBFREELISTSIZE 100
```

Size of job free list

9.159.1.13 STATS

```
#define STATS 1
```

Statistics.

Always include stats because code change is minimal.

9.159.2 Typedef Documentation

9.159.2.1 free_routine

```
typedef void(* free_routine) (void *arg)
```

Function for freeing a thread argument.

9.159.2.2 ThreadPool

```
typedef struct THREADPOOL ThreadPool
```

A thread pool similar to the thread pool in the UPnP SDK.

Allows jobs to be scheduled for running by threads in a thread pool. The thread pool is initialized with a minimum and maximum thread number as well as a max idle time and a jobs per thread ratio. If a worker thread waits the whole max idle time without receiving a job and the thread pool currently has more threads running than the minimum then the worker thread will exit. If when scheduling a job the current job to thread ratio becomes greater than the set ratio and the thread pool currently has less than the maximum threads then a new thread will be created.

9.159.2.3 ThreadPoolAttr

```
typedef struct THREADPOOLATTR ThreadPoolAttr
```

Attributes for thread pool. Used to set and change parameters of thread pool.

9.159.2.4 ThreadPoolJob

```
typedef struct THREADPOOLJOB ThreadPoolJob
```

Internal ThreadPool Job.

9.159.2.5 ThreadPoolStats

```
typedef struct TPOOLSTATS ThreadPoolStats
```

Structure to hold statistics.

9.159.3 Function Documentation

9.159.3.1 ThreadPoolAdd()

```
int ThreadPoolAdd (
            ThreadPool * tp,
            ThreadPoolJob * job,
            int * jobId )
```

Adds a job to the thread pool. Job will be run as soon as possible.

Returns

- 0 on success, nonzero on failure.
- EOUTOFMEM if not enough memory to add job.

Parameters

tp	valid thread pool pointer.
job	.
job↩ Id	id of job.

References AddWorker(), THREADPOOL::attr, THREADPOOL::condition, CreateThreadPoolJob(), FreeThreadPoolJob(), THREADPOOL::highJobQ, INVALID_JOB_ID, THREADPOOL::lastJobId, THREADPOOL::lowJobQ, THREADPOOLATTR::maxJobs THREADPOOL::medJobQ, THREADPOOL::mutex, and LINKEDLIST::size.

Referenced by genaNotifyThread(), readFromSSDPSocket(), ssdp_handle_ctrlpt_msg(), TimerThreadWorker(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.159.3.2 ThreadPoolAddPersistent()

```
int ThreadPoolAddPersistent (
            ThreadPool * tp,
            ThreadPoolJob * job,
            int * jobId )
```

Adds a persistent job to the thread pool.

Job will be run as soon as possible. Call will block until job is scheduled.

Returns

- 0 on success.
- EOUTOFMEM not enough memory to add job.
- EMAXTHREADS not enough threads to add persistent job.

Parameters

tp	Valid thread pool pointer.
job	Valid thread pool job.
job↩ Id	.

References THREADPOOL::attr, THREADPOOL::condition, CreateThreadPoolJob(), CreateWorker(), INVALID_JOB_ID, THREADPOOL::lastJobId, THREADPOOLATTR::maxThreads, THREADPOOL::mutex, THREADPOOL::persistentJob, THREADPOOL::persistentThreads, THREADPOOL::start_and_shutdown, and THREADPOOL::totalThreads.

Referenced by StartMiniServer(), TimerThreadInit(), and TimerThreadWorker().

9.159.3.3 ThreadPoolGetAttr()

```
int ThreadPoolGetAttr (
            ThreadPool * tp,
            ThreadPoolAttr * out )
```

Gets the current set of attributes associated with the thread pool.

Returns

- 0 on success, nonzero on failure.

Parameters

tp	valid thread pool pointer.
out	non null pointer to store attributes.

References THREADPOOL::attr, THREADPOOL::mutex, and THREADPOOL::shutdown.

9.159.3.4 ThreadPoolGetStats()

```
UPNP_EXPORT_SPEC int ThreadPoolGetStats (
            ThreadPool * tp,
            ThreadPoolStats * stats )
```

Returns various statistics about the thread pool.

Only valid if STATS has been defined.

Returns

Always returns 0.

Parameters

tp	Valid initialized threadpool.
stats	Valid stats, out parameter.

References THREADPOOL::highJobQ, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOL::mutex, THREADPOOL::persistentThreads, THREADPOOL::shutdown, THREADPOOL::stats, and THREADPOOL::totalThreads.

Referenced by PrintThreadPoolStats().

9.159.3.5 ThreadPoolInit()

```
int ThreadPoolInit (
            ThreadPool * tp,
            ThreadPoolAttr * attr )
```

Initializes and starts ThreadPool. Must be called first and only once for ThreadPool.

Returns

- 0 on success.
- EAGAIN if not enough system resources to create minimum threads.
- INVALID_POLICY if schedPolicy can't be set.
- EMAXTHREADS if minimum threads is greater than maximum threads.

Parameters

tp	Must be valid, non null, pointer to ThreadPool.
attr	Can be null. if not null then attr contains the following fields: • `minWorkerThreads` - minimum number of worker threads thread pool will never have less than this number of threads. • `maxWorkerThreads` - maximum number of worker threads thread pool will never have more than this number of threads. • `maxIdleTime` - maximum time that a worker thread will spend idle. If a worker is idle longer than this time and there are more than the min number of workers running, then the worker thread exits. • `jobsPerThread` - ratio of jobs to thread to try and maintain if a job is scheduled and the number of jobs per thread is greater than this number,and if less than the maximum number of workers are running then a new thread is started to help out with efficiency. • `schedPolicy` - scheduling policy to try and set (OS dependent).

References THREADPOOL::attr, THREADPOOL::busyThreads, CmpThreadPoolJob(), THREADPOOL::condition, CreateWorker(), THREADPOOL::highJobQ, INVALID_POLICY, THREADPOOL::jobFreeList, JOBFREELISTSIZE, THREADPOOL::lastJobId, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOLATTR::minThreads, THREADPOOL::mutex, THREADPOOL::pendingWorkerThreadStart, THREADPOOL::persistentJob, THREADPOOL::persistentThre THREADPOOLATTR::schedPolicy, SetPolicyType(), THREADPOOL::shutdown, THREADPOOL::start_and_shutdown, THREADPOOL::stats, StatsInit(), ThreadPoolShutdown(), THREADPOOL::totalThreads, and TPAttrInit().

Referenced by UpnpInitThreadPools().

9.159.3.6 ThreadPoolPrintStats()

```
UPNP_EXPORT_SPEC void ThreadPoolPrintStats (
            ThreadPoolStats * stats )
```

Parameters

stats	.

References StatsTime().

9.159.3.7 ThreadPoolRemove()

```
int ThreadPoolRemove (
            ThreadPool * tp,
            int jobId,
            ThreadPoolJob * out )
```

Removes a job from the thread pool. Can only remove jobs which are not currently running.

Returns

- `0` on success, nonzero on failure.
- `INVALID_JOB_ID` if job not found.

Parameters

tp	valid thread pool pointer.
job↩ Id	id of job.
out	space for removed job.

References FreeThreadPoolJob(), THREADPOOL::highJobQ, INVALID_JOB_ID, THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOL::mutex, and THREADPOOL::persistentJob.

9.159.3.8 ThreadPoolSetAttr()

```
int ThreadPoolSetAttr (
            ThreadPool * tp,
            ThreadPoolAttr * attr )
```

Sets the attributes for the thread pool. Only affects future calculations.

Returns

- `0` on success, nonzero on failure.
- `INVALID_POLICY` if policy can not be set.

Parameters

tp	valid thread pool pointer.
attr	pointer to attributes, null sets attributes to default.

References THREADPOOL::attr, THREADPOOL::condition, CreateWorker(), INVALID_POLICY, THREADPOOLATTR::minThreads, THREADPOOL::mutex, THREADPOOLATTR::schedPolicy, SetPolicyType(), ThreadPoolShutdown(), THREADPOOL::totalThreads, and TPAttrInit().

9.159.3.9 ThreadPoolShutdown()

```
int ThreadPoolShutdown (
            ThreadPool * tp )
```

Shuts the thread pool down. Waits for all threads to finish. May block indefinitely if jobs do not exit.

Returns

0 on success, nonzero on failure

Parameters

tp	must be valid tp.

References THREADPOOL::condition, FreeThreadPoolJob(), THREADPOOL::highJobQ, THREADPOOL::jobFreeList,

THREADPOOL::lowJobQ, THREADPOOL::medJobQ, THREADPOOL::mutex, THREADPOOL::persistentJob, THREADPOOL::shutdown, LINKEDLIST::size, THREADPOOL::start_and_shutdown, and THREADPOOL::totalThreads.

Referenced by ThreadPoolInit(), ThreadPoolSetAttr(), and UpnpFinish().

9.159.3.10 TPAttrInit()

```
int TPAttrInit (
            ThreadPoolAttr * attr )
```

Initializes thread pool attributes. Sets values to defaults defined in ThreadPool.h.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.

References DEFAULT_IDLE_TIME, DEFAULT_JOBS_PER_THREAD, DEFAULT_MAX_THREADS, DEFAULT_MIN_THREADS, DEFAULT_STACK_SIZE, DEFAULT_STARVATION_TIME, THREADPOOLATTR::jobsPerThread, THREADPOOLATTR::maxIdleTime, THREADPOOLATTR::maxJobsTotal, THREADPOOLATTR::maxThreads, THREADPOOLATTR::minThreads, THREADPOOLATTR::schedPolicy, THREADPOOLATTR::stackSize, and THREADPOOLATTR::starvationTime.

Referenced by ThreadPoolInit(), ThreadPoolSetAttr(), and UpnpInitThreadPools().

9.159.3.11 TPAttrSetIdleTime()

```
int TPAttrSetIdleTime (
            ThreadPoolAttr * attr,
            int idleTime )
```

Sets the idle time for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
idleTime	.

References THREADPOOLATTR::maxIdleTime.

Referenced by UpnpInitThreadPools().

9.159.3.12 TPAttrSetJobsPerThread()

```
int TPAttrSetJobsPerThread (
            ThreadPoolAttr * attr,
            int jobsPerThread )
```

Sets the jobs per thread ratio.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
jobsPerThread	number of jobs per thread to maintain.

References THREADPOOLATTR::jobsPerThread.

Referenced by UpnpInitThreadPools().

9.159.3.13 TPAttrSetMaxJobsTotal()

```
int TPAttrSetMaxJobsTotal (
            ThreadPoolAttr * attr,
            int maxJobsTotal )
```

Sets the maximum number jobs that can be qeued totally.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
maxJobsTotal	maximum number of jobs.

References THREADPOOLATTR::maxJobsTotal.

Referenced by UpnpInitThreadPools().

9.159.3.14 TPAttrSetMaxThreads()

```
int TPAttrSetMaxThreads (
            ThreadPoolAttr * attr,
            int maxThreads )
```

Sets the max threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
maxThreads	value to set.

References THREADPOOLATTR::maxThreads.

Referenced by UpnpInitThreadPools().

9.159.3.15 TPAttrSetMinThreads()

```
int TPAttrSetMinThreads (
            ThreadPoolAttr * attr,
            int minThreads )
```

Sets the min threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
minThreads	value to set.

References THREADPOOLATTR::minThreads.

Referenced by UpnpInitThreadPools().

9.159.3.16 TPAttrSetSchedPolicy()

```
int TPAttrSetSchedPolicy (
            ThreadPoolAttr * attr,
            PolicyType schedPolicy )
```

Sets the scheduling policy for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
schedPolicy	must be a valid policy type.

References THREADPOOLATTR::schedPolicy.

9.159.3.17 TPAttrSetStackSize()

```
int TPAttrSetStackSize (
            ThreadPoolAttr * attr,
            size_t stackSize )
```

Sets the stack size for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
stackSize	value to set.

References THREADPOOLATTR::stackSize.

Referenced by UpnpInitThreadPools().

9.159.3.18 TPAttrSetStarvationTime()

```
int TPAttrSetStarvationTime (
            ThreadPoolAttr * attr,
            int starvationTime )
```

Sets the starvation time for the thread pool attributes.

Returns

Always returns 0.

Parameters

attr	must be valid thread pool attributes.
starvationTime	milliseconds.

References THREADPOOLATTR::starvationTime.

9.159.3.19 TPJobInit()

```
int TPJobInit (
            ThreadPoolJob * job,
            start_routine func,
            void * arg )
```

Initializes thread pool job. Sets the priority to default defined in ThreadPool.h. Sets the free_routine to default defined in ThreadPool.h.

Returns

Always returns 0.

Parameters

job	must be valid thread pool attributes.
func	function to run, must be valid.
arg	argument to pass to function.

References DEFAULT_FREE_ROUTINE, and DEFAULT_PRIORITY.

Referenced by readFromSSDPSocket(), ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_ctrlpt_msg(), ssdp_handle_device_request(), StartMiniServer(), TimerThreadInit(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsy UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.159.3.20 TPJobSetFreeFunction()

```
int TPJobSetFreeFunction (
            ThreadPoolJob * job,
            free_routine func )
```

Sets the max threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

job	must be valid thread pool attributes.
func	value to set.

Referenced by readFromSSDPSocket(), ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_ctrlpt_msg(), ssdp_handle_device_request(), StartMiniServer(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.159.3.21 TPJobSetPriority()

```
int TPJobSetPriority (
            ThreadPoolJob * job,
            ThreadPriority priority )
```

Sets the max threads for the thread pool attributes.

Returns

Always returns 0.

Parameters

job	must be valid thread pool attributes.
priority	value to set.

Referenced by readFromSSDPSocket(), ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_ctrlpt_msg(), StartMiniServer(), TimerThreadInit(), UpnpGetServiceVarStatusAsync(), UpnpRenewSubscriptionAsync(), UpnpSendActionAsync(), UpnpSendActionExAsync(), UpnpSendAdvertisementLowPower(), UpnpSubscribeAsync(), and UpnpUnSubscribeAsync().

9.159.3.22 TPSetMaxJobsTotal()

```
void TPSetMaxJobsTotal (
             int mjt )
```

Sets the maximum number of jobs in the thread pool. This option is intended for server applications to avoid an overflow of jobs when serving e.g. many web requests.

Referenced by UpnpSetMaxJobsTotal().

9.160 ThreadPool.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  * Copyright (c) 2012 France Telecom All rights reserved.
00006  *
00007  * Redistribution and use in source and binary forms, with or without
00008  * modification, are permitted provided that the following conditions are met:
00009  *
00010  * * Redistributions of source code must retain the above copyright notice,
00011  * this list of conditions and the following disclaimer.
00012  * * Redistributions in binary form must reproduce the above copyright notice,
00013  * this list of conditions and the following disclaimer in the documentation
00014  * and/or other materials provided with the distribution.
00015  * * Neither name of Intel Corporation nor the names of its contributors
00016  * may be used to endorse or promote products derived from this software
00017  * without specific prior written permission.
00018  *
00019  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00020  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00021  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00022  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00023  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00024  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00026  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00027  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00028  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00029  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00030  *
00031  ******************************************************************************/
00032
00033 #ifndef THREADPOOL_H
00034 #define THREADPOOL_H
00035
00040 #include "FreeList.h"
00041 #include "LinkedList.h"
00042 #include "UpnpGlobal.h" /* for UPNP_INLINE, UPNP_EXPORT_SPEC */
```

```
00043 #include "UpnpInet.h"
00044 #include "ithread.h"
00045
00046 #include <errno.h>
00047
00048 #ifdef _WIN32
00049         #include <time.h>
00050
00051         #ifndef _TIMEZONE_DEFINED
00052 struct timezone
00053 {
00054         int tz_minuteswest; /* minutes W of Greenwich */
00055         int tz_dsttime;     /* type of dst correction */
00056 };
00057         #endif
00058 int gettimeofday(struct timeval *tv, struct timezone *tz);
00059 #else /* _WIN32 */
00060         #include <sys/param.h>
00061         #include <sys/time.h> /* for gettimeofday() */
00062         #if defined(__APPLE__) || defined(__NetBSD__)
00063                 #include <sys/resource.h> /* for setpriority() */
00064         #endif
00065 #endif
00066
00067 #ifdef __cplusplus
00068 extern "C" {
00069 #endif
00070
00072 #define JOBFREELISTSIZE 100
00073
00074 #define INFINITE_THREADS -1
00075
00076 #define EMAXTHREADS (-8 & 1 << 29)
00077
00079 #define INVALID_POLICY (-9 & 1 << 29)
00080
00082 #define INVALID_JOB_ID (-2 & 1 << 29)
00083
00084 typedef enum duration
00085 {
00086         SHORT_TERM,
00087         PERSISTENT
00088 } Duration;
00089
00090 typedef enum priority
00091 {
00092         LOW_PRIORITY,
00093         MED_PRIORITY,
00094         HIGH_PRIORITY
00095 } ThreadPriority;
00096
00098 #define DEFAULT_PRIORITY MED_PRIORITY
00099
00101 #define DEFAULT_MIN_THREADS 1
00102
00104 #define DEFAULT_MAX_THREADS 10
00105
00107 #define DEFAULT_STACK_SIZE 0u
00108
00110 #define DEFAULT_JOBS_PER_THREAD 10
00111
00113 #define DEFAULT_STARVATION_TIME 500
00114
00116 #define DEFAULT_IDLE_TIME 10 * 1000
00117
00119 #define DEFAULT_FREE_ROUTINE NULL
00120
00122 #define DEFAULT_MAX_JOBS_TOTAL 100
00123 extern int maxJobsTotal;
00124
00130 #define STATS 1
00131
00132 #ifdef _DEBUG
00133         #define DEBUG 1
00134 #endif
00135
00136 typedef int PolicyType;
00137
00138 #define DEFAULT_POLICY SCHED_OTHER
00139
00141 typedef void (*free_routine)(void *arg);
00142
00145 typedef struct THREADPOOLATTR
00146 {
00148         int minThreads;
00150         int maxThreads;
00152         size_t stackSize;
```

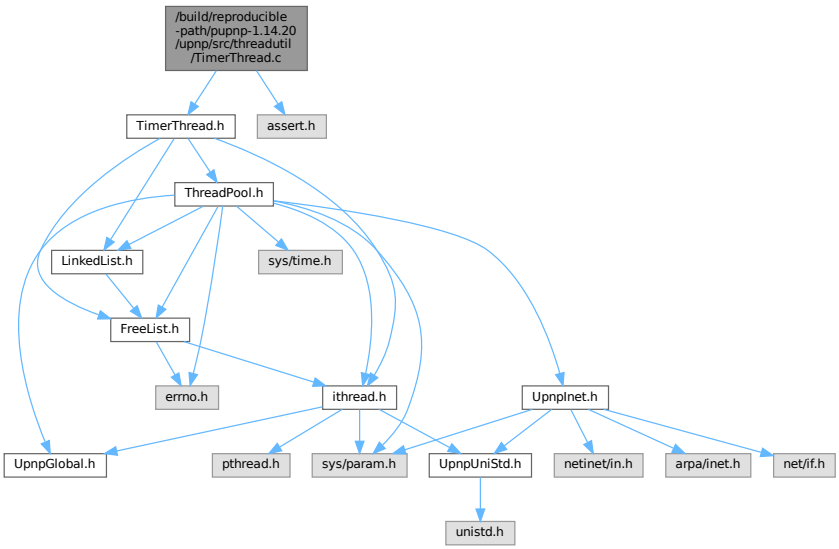
```
00155         int maxIdleTime;
00157         int jobsPerThread;
00159         int maxJobsTotal;
00162         int starvationTime;
00164         PolicyType schedPolicy;
00165 } ThreadPoolAttr;
00166
00168 typedef struct THREADPOOLJOB
00169 {
00170         start_routine func;
00171         void *arg;
00172         free_routine free_func;
00173         struct timeval requestTime;
00174         ThreadPriority priority;
00175         int jobId;
00176 } ThreadPoolJob;
00177
00179 typedef struct TPOOLSTATS
00180 {
00181         double totalTimeHQ;
00182         int totalJobsHQ;
00183         double avgWaitHQ;
00184         double totalTimeMQ;
00185         int totalJobsMQ;
00186         double avgWaitMQ;
00187         double totalTimeLQ;
00188         int totalJobsLQ;
00189         double avgWaitLQ;
00190         double totalWorkTime;
00191         double totalIdleTime;
00192         int workerThreads;
00193         int idleThreads;
00194         int persistentThreads;
00195         int totalThreads;
00196         int maxThreads;
00197         int currentJobsHQ;
00198         int currentJobsLQ;
00199         int currentJobsMQ;
00200 } ThreadPoolStats;
00201
00217 typedef struct THREADPOOL
00218 {
00220         ithread_mutex_t mutex;
00222         ithread_cond_t condition;
00224         ithread_cond_t start_and_shutdown;
00226         int lastJobId;
00228         int shutdown;
00230         int totalThreads;
00232         int pendingWorkerThreadStart;
00234         int busyThreads;
00236         int persistentThreads;
00238         FreeList jobFreeList;
00240         LinkedList lowJobQ;
00242         LinkedList medJobQ;
00244         LinkedList highJobQ;
00246         ThreadPoolJob *persistentJob;
00248         ThreadPoolAttr attr;
00250         ThreadPoolStats stats;
00251 } ThreadPool;
00252
00258 void TPSetMaxJobsTotal(int mjt);
00259
00270 int ThreadPoolInit(
00272         ThreadPool *tp,
00288         ThreadPoolAttr *attr);
00289
00300 int ThreadPoolAddPersistent(
00302         ThreadPool *tp,
00304         ThreadPoolJob *job,
00306         int *jobId);
00307
00314 int ThreadPoolGetAttr(
00316         ThreadPool *tp,
00318         ThreadPoolAttr *out);
00319
00328 int ThreadPoolSetAttr(
00330         ThreadPool *tp,
00332         ThreadPoolAttr *attr);
00333
00341 int ThreadPoolAdd(
00343         ThreadPool *tp,
00345         ThreadPoolJob *job,
00347         int *jobId);
00348
00357 int ThreadPoolRemove(
00359         ThreadPool *tp,
00361         int jobId,
```

```
00363         ThreadPoolJob *out);
00364
00371 int ThreadPoolShutdown(
00373         ThreadPool *tp);
00374
00381 int TPJobInit(
00383         ThreadPoolJob *job,
00385         start_routine func,
00387         void *arg);
00388
00394 int TPJobSetPriority(
00396         ThreadPoolJob *job,
00398         ThreadPriority priority);
00399
00405 int TPJobSetFreeFunction(
00407         ThreadPoolJob *job,
00409         free_routine func);
00410
00417 int TPAttrInit(
00419         ThreadPoolAttr *attr);
00420
00426 int TPAttrSetMaxThreads(
00428         ThreadPoolAttr *attr,
00430         int maxThreads);
00431
00437 int TPAttrSetMinThreads(
00439         ThreadPoolAttr *attr,
00441         int minThreads);
00442
00448 int TPAttrSetStackSize(
00450         ThreadPoolAttr *attr,
00452         size_t stackSize);
00453
00459 int TPAttrSetIdleTime(
00461         ThreadPoolAttr *attr,
00463         int idleTime);
00464
00470 int TPAttrSetJobsPerThread(
00472         ThreadPoolAttr *attr,
00474         int jobsPerThread);
00475
00481 int TPAttrSetStarvationTime(
00483         ThreadPoolAttr *attr,
00485         int starvationTime);
00486
00492 int TPAttrSetSchedPolicy(
00494         ThreadPoolAttr *attr,
00496         PolicyType schedPolicy);
00497
00503 int TPAttrSetMaxJobsTotal(
00505         ThreadPoolAttr *attr,
00507         int maxJobsTotal);
00508
00516 #ifdef STATS
00517 UPNP_EXPORT_SPEC int ThreadPoolGetStats(
00519         ThreadPool *tp,
00521         ThreadPoolStats *stats);
00522 #else
00523 static UPNP_INLINE int ThreadPoolGetStats(
00525         ThreadPool *tp,
00527         ThreadPoolStats *stats)
00528 {
00529 }
00530 #endif
00531
00535 #ifdef STATS
00536 UPNP_EXPORT_SPEC void ThreadPoolPrintStats(
00538         ThreadPoolStats *stats);
00539 #else
00540 static UPNP_INLINE void ThreadPoolPrintStats(
00542         ThreadPoolStats *stats)
00543 {
00544 }
00545 #endif
00546
00547 #ifdef __cplusplus
00548 }
00549 #endif
00550
00551 #endif /* THREADPOOL_H */
```

9.161 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.c File Reference

```
#include "TimerThread.h"
#include <assert.h>
```

Include dependency graph for TimerThread.c:



Functions

- static void FreeTimerEvent (TimerThread *timer, TimerEvent *event)

  Deallocates a dynamically allocated TimerEvent.
- static void TimerThreadWorker (void *arg)

  Implements timer thread.
- static int CalculateEventTime (time_t *timeout, TimeoutType type)

  Calculates the appropriate timeout in absolute seconds since Jan 1, 1970.
- static TimerEvent * CreateTimerEvent (TimerThread *timer, ThreadPoolJob *job, Duration persistent, time_t eventTime, int id)

  Creates a Timer Event. (Dynamically allocated).
- int TimerThreadInit (TimerThread *timer, ThreadPool *tp)

  Initializes and starts timer thread.
- int TimerThreadSchedule (TimerThread *timer, time_t timeout, TimeoutType type, ThreadPoolJob *job, Duration duration, int *id)

  Schedules an event to run at a specified time.
- int TimerThreadRemove (TimerThread *timer, int id, ThreadPoolJob *out)

  Removes an event from the timer Q.
- int TimerThreadShutdown (TimerThread *timer)

  Shutdown the timer thread.

9.161.1 Function Documentation

9.161.1.1 CalculateEventTime()

```
static int CalculateEventTime (
            time_t * timeout,
            TimeoutType type )  [static]
```

Calculates the appropriate timeout in absolute seconds since Jan 1, 1970.

Returns

Parameters

in	*timeout*	Timeout.
in	*type*	Timeout type.

References ABS_SEC.

Referenced by TimerThreadSchedule().

9.161.1.2 CreateTimerEvent()

```
static TimerEvent * CreateTimerEvent (
            TimerThread * timer,
            ThreadPoolJob * job,
            Duration persistent,
            time_t eventTime,
            int id )  [static]
```

Creates a Timer Event. (Dynamically allocated).

Returns

(TimerEvent ∗) on success, NULL on failure.

Parameters

in	*timer*	Valid timer thread pointer.
in	*job*	.
in	*persistent*	.
in	*eventTime*	The absoule time of the event in seconds from Jan, 1970.
in	*id*	Id of job.

References TIMEREVENT::eventTime, and TIMEREVENT::persistent.

Referenced by TimerThreadSchedule().

9.161.1.3 FreeTimerEvent()

```
static void FreeTimerEvent (
             TimerThread * timer,
             TimerEvent * event )  [static]
```

Deallocates a dynamically allocated TimerEvent.

Parameters

in	*timer*	Valid timer thread pointer.
in	*event*	Must be allocated with CreateTimerEvent

Referenced by TimerThreadRemove(), TimerThreadSchedule(), TimerThreadShutdown(), and TimerThreadWorker().

9.161.1.4 TimerThreadInit()

```
int TimerThreadInit (
             TimerThread * timer,
             ThreadPool * tp )
```

Initializes and starts timer thread.

Returns

0 on success, nonzero on failure. Returns error from ThreadPoolAddPersistent on failure.

Parameters

in	*timer*	Valid timer thread pointer.
in	*tp*	Valid thread pool to use. Must be started. Must be valid for lifetime of timer. Timer must be shutdown BEFORE thread pool.

References ThreadPoolAddPersistent(), TimerThreadWorker(), TPJobInit(), and TPJobSetPriority().

Referenced by UpnpInitPreamble().

9.161.1.5 TimerThreadRemove()

```
int TimerThreadRemove (
             TimerThread * timer,
             int id,
             ThreadPoolJob * out )
```

Removes an event from the timer Q.

Events can only be removed before they have been placed in the thread pool.

Returns

0 on success, INVALID_EVENT_ID on failure.

Parameters

in	*timer*	Valid timer thread pointer.
in	*id*	Id of event to remove.
in	*out*	Space for thread pool job.

References FreeTimerEvent().

Referenced by free_client_subscription(), and genaRenewSubscription().

9.161.1.6 TimerThreadSchedule()

```
int TimerThreadSchedule (
            TimerThread * timer,
            time_t time,
            TimeoutType type,
            ThreadPoolJob * job,
            Duration duration,
            int * id )
```

Schedules an event to run at a specified time.

Returns

0 on success, nonzero on failure, EOUTOFMEM if not enough memory to schedule job.

Parameters

in	*timer*	Valid timer thread pointer.
in	*timeout*	time of event. Either in absolute seconds, or relative seconds in the future.
in	*type*	either ABS_SEC, or REL_SEC. If REL_SEC, then the event will be scheduled at the current time + REL_SEC.
in	*job*	Valid Thread pool job with following fields.
in	*duration*	.
in	*id*	Id of timer event. (out, can be null).

References CalculateEventTime(), CreateTimerEvent(), TIMEREVENT::eventTime, and FreeTimerEvent().

Referenced by ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_device_request(), and UpnpSendAdvertisementLowPow

9.161.1.7 TimerThreadShutdown()

```
int TimerThreadShutdown (
            TimerThread * timer )
```

Shutdown the timer thread.

Events scheduled in the future will NOT be run.

Timer thread should be shutdown BEFORE it's associated thread pool.

Returns

0 if succesfull, nonzero otherwise. Always returns 0.

Parameters

in	*timer*	Valid timer thread pointer.

References FreeTimerEvent().

Referenced by UpnpFinish().

9.161.1.8 TimerThreadWorker()

```
static void TimerThreadWorker (
            void * arg )  [static]
```

Implements timer thread.

Waits for next event to occur and schedules associated job into threadpool.

Parameters

in	*arg*	arg is cast to (TimerThread *).

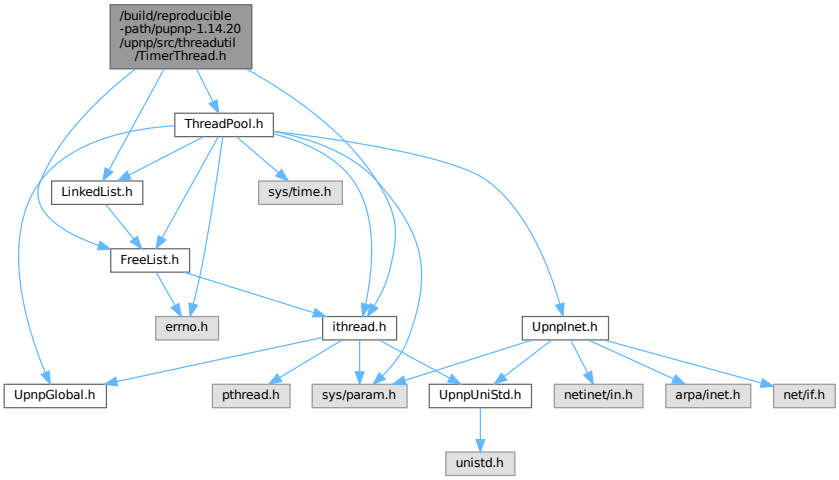
References TIMEREVENT::eventTime, FreeTimerEvent(), TIMEREVENT::persistent, LINKEDLIST::size, ThreadPoolAdd(), and ThreadPoolAddPersistent().

Referenced by TimerThreadInit().

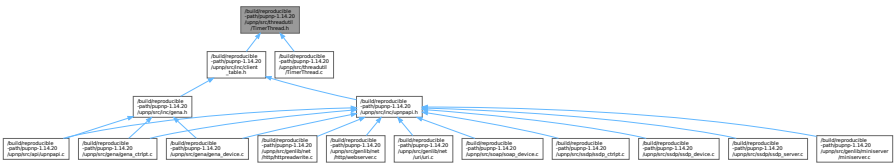
9.162 /build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.h File Reference

```
#include "FreeList.h"
#include "LinkedList.h"
#include "ThreadPool.h"
#include "ithread.h"
```

Include dependency graph for TimerThread.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct TIMERTHREAD
- struct TIMEREVENT

Macros

- #define **INVALID_EVENT_ID** (-10 & 1 << 29)

Typedefs

- typedef enum timeoutType TimeoutType
- typedef struct TIMERTHREAD TimerThread
- typedef struct TIMEREVENT TimerEvent

Enumerations

- enum timeoutType { ABS_SEC , REL_SEC }

Functions

- int TimerThreadInit (TimerThread *timer, ThreadPool *tp)

  Initializes and starts timer thread.
- int TimerThreadSchedule (TimerThread *timer, time_t time, TimeoutType type, ThreadPoolJob *job, Duration duration, int *id)

  Schedules an event to run at a specified time.
- int TimerThreadRemove (TimerThread *timer, int id, ThreadPoolJob *out)

  Removes an event from the timer Q.
- int TimerThreadShutdown (TimerThread *timer)

  Shutdown the timer thread.

9.162.1 Typedef Documentation

9.162.1.1 TimeoutType

```
typedef enum timeoutType TimeoutType
```

Timeout Types.

9.162.1.2 TimerEvent

```
typedef struct TIMEREVENT TimerEvent
```

Struct to contain information for a timer event.

Internal to the TimerThread.

9.162.1.3 TimerThread

```
typedef struct TIMERTHREAD TimerThread
```

A timer thread similar to the one in the Upnp SDK that allows the scheduling of a job to run at a specified time in the future.

Because the timer thread uses the thread pool there is no gurantee of timing, only approximate timing.

Uses ThreadPool, Mutex, Condition, Thread.

9.162.2 Enumeration Type Documentation

9.162.2.1 timeoutType

```
enum timeoutType
```

Timeout Types.

Enumerator

ABS_SEC	seconds from Jan 1, 1970.
REL_SEC	seconds from current time.

9.162.3 Function Documentation

9.162.3.1 TimerThreadInit()

```
int TimerThreadInit (
            TimerThread * timer,
            ThreadPool * tp )
```

Initializes and starts timer thread.

Returns

0 on success, nonzero on failure. Returns error from ThreadPoolAddPersistent on failure.

Parameters

in	*timer*	Valid timer thread pointer.
in	*tp*	Valid thread pool to use. Must be started. Must be valid for lifetime of timer. Timer must be shutdown BEFORE thread pool.

References ThreadPoolAddPersistent(), TimerThreadWorker(), TPJobInit(), and TPJobSetPriority().

Referenced by UpnpInitPreamble().

9.162.3.2 TimerThreadRemove()

```
int TimerThreadRemove (
            TimerThread * timer,
            int id,
            ThreadPoolJob * out )
```

Removes an event from the timer Q.

Events can only be removed before they have been placed in the thread pool.

Returns

0 on success, INVALID_EVENT_ID on failure.

Parameters

in	*timer*	Valid timer thread pointer.
in	*id*	Id of event to remove.
in	*out*	Space for thread pool job.

References FreeTimerEvent().

Referenced by free_client_subscription(), and genaRenewSubscription().

9.162.3.3 TimerThreadSchedule()

```
int TimerThreadSchedule (
              TimerThread * timer,
              time_t time,
              TimeoutType type,
              ThreadPoolJob * job,
              Duration duration,
              int * id )
```

Schedules an event to run at a specified time.

Returns

0 on success, nonzero on failure, EOUTOFMEM if not enough memory to schedule job.

Parameters

in	*timer*	Valid timer thread pointer.
in	*time*	time of event. Either in absolute seconds, or relative seconds in the future.
in	*type*	either ABS_SEC, or REL_SEC. If REL_SEC, then the event will be scheduled at the current time + REL_SEC.
in	*job*	Valid Thread pool job with following fields.
in	*duration*	.
in	*id*	Id of timer event. (out, can be null).

References CalculateEventTime(), CreateTimerEvent(), TIMEREVENT::eventTime, and FreeTimerEvent().

Referenced by ScheduleGenaAutoRenew(), SearchByTarget(), ssdp_handle_device_request(), and UpnpSendAdvertisementLowPow

9.162.3.4 TimerThreadShutdown()

```
int TimerThreadShutdown (
              TimerThread * timer )
```

Shutdown the timer thread.

Events scheduled in the future will NOT be run.

Timer thread should be shutdown BEFORE it's associated thread pool.

Returns

0 if succesfull, nonzero otherwise. Always returns 0.

Parameters

in	*timer*	Valid timer thread pointer.

References FreeTimerEvent().

Referenced by UpnpFinish().

9.163 TimerThread.h

Go to the documentation of this file.

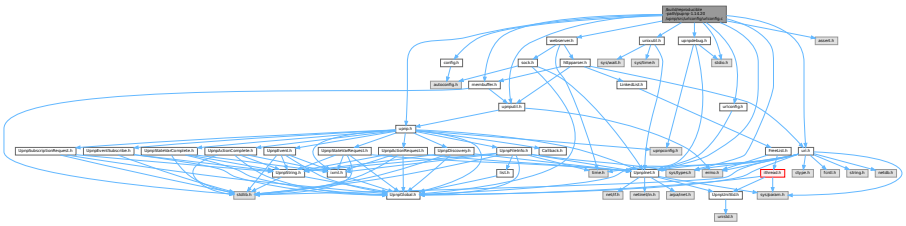
```
00001 /*******************************************************************************
00002  *
00003  * Copyright (c) 2000-2003 Intel Corporation
00004  * All rights reserved.
00005  *
00006  * Redistribution and use in source and binary forms, with or without
00007  * modification, are permitted provided that the following conditions are met:
00008  *
00009  * * Redistributions of source code must retain the above copyright notice,
00010  * this list of conditions and the following disclaimer.
00011  * * Redistributions in binary form must reproduce the above copyright notice,
00012  * this list of conditions and the following disclaimer in the documentation
00013  * and/or other materials provided with the distribution.
00014  * * Neither name of Intel Corporation nor the names of its contributors
00015  * may be used to endorse or promote products derived from this software
00016  * without specific prior written permission.
00017  *
00018  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR
00022  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00023  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00024  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
00025  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
00026  * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
00027  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00028  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029  *
00030  ******************************************************************************/
00031
00032 #ifndef TIMERTHREAD_H
00033 #define TIMERTHREAD_H
00034
00039 #include "FreeList.h"
00040 #include "LinkedList.h"
00041 #include "ThreadPool.h"
00042 #include "ithread.h"
00043
00044 #ifdef __cplusplus
00045 extern "C" {
00046 #endif
00047
00048 #define INVALID_EVENT_ID (-10 & 1 « 29)
00049
00051 typedef enum timeoutType
00052 {
00054         ABS_SEC,
00056         REL_SEC
00057 } TimeoutType;
00058
00068 typedef struct TIMERTHREAD
00069 {
00070         ithread_mutex_t mutex;
00071         ithread_cond_t condition;
00072         int lastEventId;
00073         LinkedList eventQ;
00074         int shutdown;
00075         FreeList freeEvents;
00076         ThreadPool *tp;
00077 } TimerThread;
00078
00084 typedef struct TIMEREVENT
00085 {
00086         ThreadPoolJob job;
```

```
00088         time_t eventTime;
00090         Duration persistent;
00091         int id;
00092 } TimerEvent;
00093
00100 int TimerThreadInit(
00102         TimerThread *timer,
00105         ThreadPool *tp);
00106
00113 int TimerThreadSchedule(
00115         TimerThread *timer,
00118         time_t time,
00121         TimeoutType type,
00123         ThreadPoolJob *job,
00125         Duration duration,
00127         int *id);
00128
00136 int TimerThreadRemove(
00138         TimerThread *timer,
00140         int id,
00142         ThreadPoolJob *out);
00143
00153 int TimerThreadShutdown(
00155         TimerThread *timer);
00156
00157 #ifdef __cplusplus
00158 }
00159 #endif
00160
00161 #endif /* TIMER_THREAD_H */
```

9.164 /build/reproducible-path/pupnp-1.14.20/upnp/src/urlconfig/urlconfig.c File Reference

```
#include "config.h"
#include "UpnpInet.h"
#include "membuffer.h"
#include "unixutil.h"
#include "upnp.h"
#include "upnpdebug.h"
#include "upnputil.h"
#include "uri.h"
#include "urlconfig.h"
#include "webserver.h"
#include <assert.h>
#include <stdio.h>
#include <sys/types.h>
```

Include dependency graph for urlconfig.c:

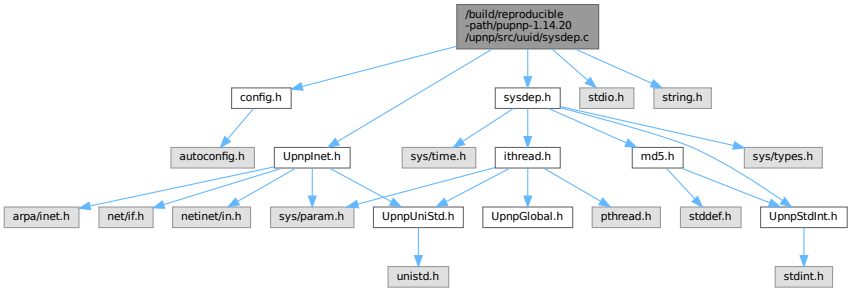


9.165 /build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/sysdep.c File Reference

```
#include "config.h"
#include "sysdep.h"
```

```
#include "UpnpInet.h"
#include <stdio.h>
#include <string.h>
```

Include dependency graph for sysdep.c:



Functions

- void get_ieee_node_identifier (uuid_node_t *node)

  System dependent call to get IEEE node ID.
- void get_system_time (uuid_time_t *uuid_time)

  System dependent call to get the current system time.
- void **get_random_info** (unsigned char seed[16])

9.165.1 Function Documentation

9.165.1.1 get_ieee_node_identifier()

```
void get_ieee_node_identifier (
            uuid_node_t * node )
```

System dependent call to get IEEE node ID.

This sample implementation generates a random node ID.

Referenced by uuid_create().

9.165.1.2 get_system_time()

```
void get_system_time (
            uuid_time_t * uuid_time )
```

System dependent call to get the current system time.

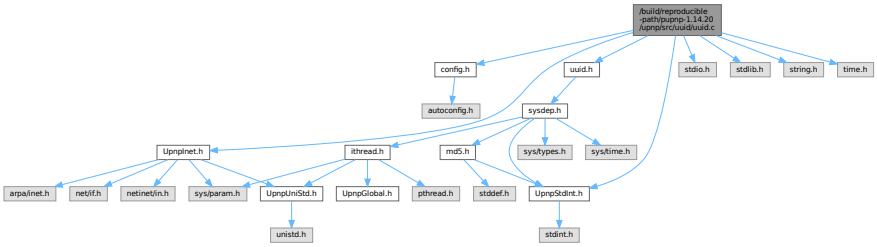
Returned as 100ns ticks since Oct 15, 1582, but resolution may be less than 100ns.

Referenced by get_current_time(), and true_random().

9.166 /build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/uuid.c File Reference

```
#include "config.h"
#include "uuid.h"
#include "UpnpInet.h"
#include "UpnpStdInt.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

Include dependency graph for uuid.c:



Data Structures

- struct uuid_state

Macros

- #define CHECK(f1, f2)

Functions

- static int read_state (uint16_t *clockseq, uuid_time_t *timestamp, uuid_node_t *node)

  Read UUID generator state from non-volatile store.
- static void write_state (uint16_t clockseq, uuid_time_t timestamp, uuid_node_t node)

  Save UUID generator state back to non-volatile storage.
- static void format_uuid_v1 (uuid_upnp *uid, uint16_t clock_seq, uuid_time_t timestamp, uuid_node_t node)

  Make a UUID from the timestamp, clockseq, and node ID.
- static void format_uuid_v3 (uuid_upnp *uid, unsigned char hash[16])

  Make a UUID from a (pseudo)random 128 bit number.
- static void get_current_time (uuid_time_t *timestamp)

  Get time as 60 bit 100ns ticks since whenever.
- static uint16_t true_random (void)

  generate a crypto-quality random number. This sample doesn't do that.
- int uuid_create (uuid_upnp *uid)

  Generator of a UUID.
- void upnp_uuid_unpack (uuid_upnp *u, char *out)

  Out will be xxxx-xx-xx-xx-xxxxxx format.
- void uuid_create_from_name (uuid_upnp *uid, uuid_upnp nsid, void *name, int namelen)

  Create a UUID using a "name" from a "name space".
- int uuid_compare (uuid_upnp *u1, uuid_upnp *u2)

  Compare two UUID's "lexically" and return.

Variables

- static uuid_state **st**
- static int **stateInited** = 0

9.166.1 Macro Definition Documentation

9.166.1.1 CHECK

```
#define CHECK(
               f1,
               f2 )
```

Value:

```
        if (f1 != f2) \
                return f1 < f2 ? -1 : 1;
```

9.166.2 Function Documentation

9.166.2.1 format_uuid_v1()

```
void format_uuid_v1 (
             uuid_upnp * uid,
             uint16_t clockseq,
             uuid_time_t timestamp,
             uuid_node_t node )  [static]
```

Make a UUID from the timestamp, clockseq, and node ID.

References _uuid_upnp::clock_seq_hi_and_reserved, _uuid_upnp::clock_seq_low, _uuid_upnp::node, _uuid_upnp::time_hi_and_ver
_uuid_upnp::time_low, and _uuid_upnp::time_mid.

Referenced by uuid_create().

9.166.2.2 format_uuid_v3()

```
void format_uuid_v3 (
             uuid_upnp * uid,
             unsigned char hash[16] )  [static]
```

Make a UUID from a (pseudo)random 128 bit number.

References _uuid_upnp::clock_seq_hi_and_reserved, _uuid_upnp::time_hi_and_version, _uuid_upnp::time_low, and _uuid_upnp::time_mid.

Referenced by uuid_create_from_name().

9.166.2.3 get_current_time()

```
void get_current_time (
            uuid_time_t * timestamp )  [static]
```

Get time as 60 bit 100ns ticks since whenever.

Compensate for the fact that real clock resolution is less than 100ns.

References get_system_time(), and UUIDS_PER_TICK.

Referenced by uuid_create().

9.166.2.4 read_state()

```
int read_state (
            uint16_t * clockseq,
            uuid_time_t * timestamp,
            uuid_node_t * node )  [static]
```

Read UUID generator state from non-volatile store.

References uuid_state::cs, uuid_state::node, and uuid_state::ts.

Referenced by uuid_create().

9.166.2.5 true_random()

```
static uint16_t true_random (
            void  )  [static]
```

generate a crypto-quality random number. This sample doesn't do that.

References get_system_time(), and UUIDS_PER_TICK.

Referenced by uuid_create().

9.166.2.6 upnp_uuid_unpack()

```
void upnp_uuid_unpack (
            uuid_upnp * u,
            char * out )
```

Out will be xxxx-xx-xx-xx-xxxxxx format.

Parameters

u	.
out	.

References _uuid_upnp::clock_seq_hi_and_reserved, _uuid_upnp::clock_seq_low, _uuid_upnp::node, _uuid_upnp::time_hi_and_ver

_uuid_upnp::time_low, and _uuid_upnp::time_mid.

9.166.2.7 uuid_compare()

```
int uuid_compare (
             uuid_upnp * u1,
             uuid_upnp * u2 )
```

Compare two UUID's "lexically" and return.

Compare two UUID's "lexically".

- -1: u1 is lexically before u2
- 0: u1 is equal to u2
- 1: u1 is lexically after u2

Note: Lexical ordering is not temporal ordering!

Parameters

u1	.
u2	.

References _uuid_upnp::clock_seq_hi_and_reserved, _uuid_upnp::clock_seq_low, _uuid_upnp::node, _uuid_upnp::time_hi_and_ver
_uuid_upnp::time_low, and _uuid_upnp::time_mid.

9.166.2.8 uuid_create()

```
int uuid_create (
             uuid_upnp * uid )
```

Generator of a UUID.

Generate a UUID.

Parameters

uid	.

References format_uuid_v1(), get_current_time(), get_ieee_node_identifier(), read_state(), true_random(), and write_state().

9.166.2.9 uuid_create_from_name()

```
void uuid_create_from_name (
             uuid_upnp * uid,
             uuid_upnp nsid,
```

```
            void * name,
            int namelen )
```

Create a UUID using a "name" from a "name space".

Parameters

uid	resulting UUID.
nsid	UUID to serve as context, so identical names from different name spaces generate different UUIDs.
name	The name from which to generate a UUID.
namelen	The length of the name.

References format_uuid_v3(), _uuid_upnp::time_hi_and_version, _uuid_upnp::time_low, and _uuid_upnp::time_mid.

9.166.2.10 write_state()

```
void write_state (
            uint16_t clockseq,
            uuid_time_t timestamp,
            uuid_node_t node )  [static]
```

Save UUID generator state back to non-volatile storage.

References uuid_state::cs, uuid_state::node, and uuid_state::ts.

Referenced by uuid_create().

Index

/build/reproducible-path/pupnp-1.14.20/ixml/inc/ixml.h, 301, 308
/build/reproducible-path/pupnp-1.14.20/ixml/inc/ixmldebug.h, 314, 316
/build/reproducible-path/pupnp-1.14.20/ixml/inc/posix_overwrites.h, 317
/build/reproducible-path/pupnp-1.14.20/ixml/src/document.c, 318
/build/reproducible-path/pupnp-1.14.20/ixml/src/element.c, 319
/build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlmembuf.h, 322, 326
/build/reproducible-path/pupnp-1.14.20/ixml/src/inc/ixmlparser.h, 328, 335
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixml.c, 337
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixmldebug.c, 340
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlmembuf.c, 342
/build/reproducible-path/pupnp-1.14.20/ixml/src/ixmlparser.c, 347
/build/reproducible-path/pupnp-1.14.20/ixml/src/namedNodeMap.c, 371
/build/reproducible-path/pupnp-1.14.20/ixml/src/node.c, 373
/build/reproducible-path/pupnp-1.14.20/ixml/src/nodeList.c, 384
/build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.c, 387
/build/reproducible-path/pupnp-1.14.20/upnp/generator/TestClass.h, 394, 401
/build/reproducible-path/pupnp-1.14.20/upnp/generator/TheStruct.h, 402
/build/reproducible-path/pupnp-1.14.20/upnp/generator/generator.h, 386
/build/reproducible-path/pupnp-1.14.20/upnp/inc/Callback.h, 402, 405
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionComplete.h, 435, 440
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpActionRequest.h, 441, 453
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpDiscovery.h, 461, 477
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEvent.h, 479, 483
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpEventSubscribe.h, 484, 490
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpExtraHeaders.h, 491, 498
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpFileInfo.h, 499, 506
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpGlobal.h, 507, 509
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpInet.h, 510, 511
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpIntTypes.h, 512
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarComplete.h, 512, 519
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStateVarRequest.h, 519, 531
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpStdInt.h, 532
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpString.h, 532, 534
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpSubscriptionReques
/build/reproducible-path/pupnp-1.14.20/upnp/inc/UpnpUniStd.h, 535, 542
/build/reproducible-path/pupnp-1.14.20/upnp/inc/ithread.h, 546
/build/reproducible-path/pupnp-1.14.20/upnp/inc/list.h, 406, 408
/build/reproducible-path/pupnp-1.14.20/upnp/inc/upnp.h, 418
/build/reproducible-path/pupnp-1.14.20/upnp/inc/upnpdebug.h, 419, 426
/build/reproducible-path/pupnp-1.14.20/upnp/inc/upnptools.h, 455, 459
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/sample_util
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.c, 543, 545
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_ctrlpt.h, 546
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.c, 547, 549
/build/reproducible-path/pupnp-1.14.20/upnp/sample/common/tv_device.h, 551
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionComplete
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpActionRequest.c, 553, 556
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpDiscovery.c, 558
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEvent.c, 560, 563
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEventSubscribe.c, 567
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpExtraHeaders.c, 572
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpFileInfo.c, 594

/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEvent.c, 610
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpEventSubscribe.c, 614
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpExtraHeaders.c, 620
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpFileInfo.c, 626
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarComplete.c, 633
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpStateVarRequest.c, 639
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpString.c, 650
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/UpnpSubscriptionRequest.c, 651
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpapi.c, 584
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnpdebug.c, 590
/build/reproducible-path/pupnp-1.14.20/upnp/src/api/upnptools.c, 658
/build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_ctrlpt.c, 661
/build/reproducible-path/pupnp-1.14.20/upnp/src/gena/gena_device.c, 667
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/client_table/GenlibClientSubscription.c, 677
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/miniserver/miniserver.c, 685
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/httpreadwrite.c, 690
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/parsetools.c, 702
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/http/webserver.c, 703
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/sock.c, 718
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/net/uri/uri.c, 719
/build/reproducible-path/pupnp-1.14.20/upnp/src/genlib/util/util.c, 729
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/GenlibClientSubscription.h, 753, 762
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/VirtualDir.h, 843
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/client_table.h, 731, 734
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/config.h, 735
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena.h, 737, 746
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_ctrlpt.h, 749, 750
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/gena_device.h, 750, 753
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpparser.h, 763, 766
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/httpreadwrite.h, 771
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/inet_pton.h, 775
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/md5.h, 775
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/membuffer.h, 776, 783
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/miniserver.h, 784, 788
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/netall.h, 789
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/parsetools.h, 789, 790
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/service_table.h, 791, 800
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/soaplib.h, 803, 804
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sock.h, 806, 807
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/ssdplib.h, 809, 811
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/statcodes.h, 816
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/strintmap.h, 817
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/sysdep.h, 818, 821
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/unixutil.h, 821
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnpapi.h, 822, 825
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/upnputil.h, 828, 830
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uri.h, 831, 839
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/urlconfig.h, 841
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/uuid.h, 842
/build/reproducible-path/pupnp-1.14.20/upnp/src/inc/webserver.h, 843
/build/reproducible-path/pupnp-1.14.20/upnp/src/inet_pton.c, 844
/build/reproducible-path/pupnp-1.14.20/upnp/src/soap/soap_device.c, 844
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.c, 855
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultData.h, 859, 863
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultDataCa
863
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/SSDPResultDataCa
864, 866
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_ctrlpt.c, 851
/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_device.c, 852

/build/reproducible-path/pupnp-1.14.20/upnp/src/ssdp/ssdp_server.c, 854
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/FreeList.h, 866, 869
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/LinkedList.h, 870, 878
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.c, 879
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/ThreadPool.h, 898, 913
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.c, 917
/build/reproducible-path/pupnp-1.14.20/upnp/src/threadutil/TimerThread.h, 921, 926
/build/reproducible-path/pupnp-1.14.20/upnp/src/urlconfig/urlconfig.c, 927
/build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/sysdep.c, 927
/build/reproducible-path/pupnp-1.14.20/upnp/src/uuid/uuid.c, 929
_IXML_ATTR, 223
_IXML_CDATASection, 224
_IXML_Document, 225
_IXML_Element, 225
_IXML_ElementStack, 226
_IXML_NamedNodeMap, 227
_IXML_NamespaceURI, 228
_IXML_Node, 228
_IXML_NodeList, 229
_IXML_Text, 230
_Parser, 231
  curPtr, 231
  dataBuffer, 231
  savePtr, 231
_uuid_upnp, 232
  clock_seq_hi_and_reserved, 232
  clock_seq_low, 232
  node, 232
  time_hi_and_version, 232
  time_low, 233
  time_mid, 233

ABS_SEC
  TimerThread.h, 924
ActionNames
  TvService, 289
actions
  TvService, 289
addServiceTable
  service_table.h, 793
addToAction
  upnptools.c, 659
AddWorker
  ThreadPool.c, 882
AdvertiseAndReply
  SSDP Library, 165
advertiseAndReplyThread
  SSDP Library, 165
alias_grab
  webserver.c, 706
alias_release
  webserver.c, 706
aliasInstalled
  Handle_Info, 238
AllocGenaHeaders
  gena_device.c, 668
amount_discarded
  http_message_t, 243
API changes after 1.10, 1
APPLICATION_LISTENING_PORT
  miniserver.c, 686
APPLICATION_PORT
  THREADPOOL, 278
AutoAdvertise
  UPnP API, 36

buf
  membuffer, 253
  memptr, 255
BumpPriority
  ThreadPool.c, 882
busyThreads
  THREADPOOL, 278
bWebServerState
  UPnP API, 91

CalculateEventTime
  TimerThread.c, 918
CalcWaitTime
  ThreadPool.c, 882
Callback
  Handle_Info, 238
Callback.h
  UPNP_CONTROL_ACTION_COMPLETE, 404
  UPNP_CONTROL_ACTION_REQUEST, 404
  UPNP_CONTROL_GET_VAR_COMPLETE, 404
  UPNP_CONTROL_GET_VAR_REQUEST, 404
  UPNP_DISCOVERY_ADVERTISEMENT_ALIVE, 404
  UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE, 404
  UPNP_DISCOVERY_SEARCH_RESULT, 404
  UPNP_DISCOVERY_SEARCH_TIMEOUT, 404
  UPNP_EVENT_AUTORENEWAL_FAILED, 405
  UPNP_EVENT_RECEIVED, 404
  UPNP_EVENT_RENEWAL_COMPLETE, 405
  UPNP_EVENT_SUBSCRIBE_COMPLETE, 405
  UPNP_EVENT_SUBSCRIPTION_EXPIRED, 405
  UPNP_EVENT_SUBSCRIPTION_REQUEST, 404
  UPNP_EVENT_UNSUBSCRIBE_COMPLETE, 405
  Upnp_EventType_e, 403
  Upnp_FunPtr, 403
capacity
  membuffer, 253
char_info, 233
CHECK
  uuid.c, 930

Check_Connect_And_Wait_Connection
httpreadwrite.c, 692
check_soap_request
soap_device.c, 846
check_soapaction_hdr
soap_device.c, 847
CheckOtherHTTPHeaders
webserver.c, 707
client_table.h
free_client_subscription, 732
freeClientSubList, 732
GetClientSubActualSID, 733
GetClientSubClientSID, 733
RemoveClientSubClientSID, 733
ClientSubList
Handle_Info, 238
clock_seq_hi_and_reserved
_uuid_upnp, 232
clock_seq_low
_uuid_upnp, 232
close
VirtualDirCallbacks, 297
cmdloop_cmdlist
Sample Code, 149
cmdloop_commands, 233
cmdloop_tvcmds
Sample Code, 124
cmp_func
LINKEDLIST, 251
cmp_routine
LinkedList.h, 872
CmpThreadPoolJob
ThreadPool.c, 883
condition
THREADPOOL, 278
connfd
mserv_request_t, 255
content_subtype
document_type_t, 234
content_type
document_type_t, 234
Cookie
Handle_Info, 238
SendInstruction, 268
copy_subscription
service_table.h, 793
copy_token
uri.c, 721
copy_URL_list
uri.c, 721
uri.h, 833
copy_with_escape
ixml.c, 338
CorsHeader
SendInstruction, 268
create_ssdp_sock_reqv4
SSDP Library, 165
create_ssdp_sock_v4
SSDP Library, 166
create_url_list
gena_device.c, 669
CreateClientRequestPacket
SSDP Library, 166
CreateHTTPRangeResponseHeader
webserver.c, 707
CreateServicePacket
SSDP Library, 166
CreateThreadPoolJob
ThreadPool.c, 883
CreateTimerEvent
TimerThread.c, 918
CreateWorker
ThreadPool.c, 884
cs
uuid_state, 295
ct
xml_alias_t, 299
curPtr
_Parser, 231
cursor
scanner_t, 267

dataBuffer
_Parser, 231
default_advr_expire
Sample Code, 149
DEFAULT_FREE_ROUTINE
ThreadPool.h, 901
DEFAULT_IDLE_TIME
ThreadPool.h, 901
DEFAULT_JOBS_PER_THREAD
ThreadPool.h, 901
DEFAULT_MAX_JOBS_TOTAL
ThreadPool.h, 901
DEFAULT_MAX_THREADS
ThreadPool.h, 901
DEFAULT_MIN_THREADS
ThreadPool.h, 901
DEFAULT_PRIORITY
ThreadPool.h, 901
DEFAULT_STACK_SIZE
ThreadPool.h, 901
DEFAULT_STARVATION_TIME
ThreadPool.h, 902
default_timeout
Sample Code, 149
Deprecated List, 3
DescDocument
Handle_Info, 238
DescURL
Handle_Info, 238
DescXML
Handle_Info, 239
device_handle
Sample Code, 149
device_main
Sample Code, 125

DeviceAdvertisement
SSDP Library, 167
DeviceAf
Handle_Info, 239
DeviceList
Handle_Info, 239
DeviceListMutex
Sample Code, 150
DeviceReply
SSDP Library, 167
DeviceShutdown
SSDP Library, 168
DiffMillis
ThreadPool.c, 884
display_mutex
Sample Code, 150
doc
xml_alias_t, 299
document.c
ixmlDocument_setOwnerDocument, 319
document_type_t, 234
content_subtype, 234
content_type, 234
file_ext, 234
DOMString
XML API, 183

element.c
ixmlElement_findAttributeNode, 321
ixmlElement_setTagName, 321
entire_msg_loaded
scanner_t, 267
entity
http_message_t, 243
entity_start_position
http_parser_t, 246
error_respond
gena.h, 740
ErrorMessages
upnptools.c, 660
ErrorString, 234
rc, 235
rcError, 235
eventTime
TIMEREVENT, 284
extractIPv6address
SSDP Library, 168
ExtraHTTPHeaders
webserver.c, 708

file_ext
document_type_t, 234
FindServiceControlURLPath
service_table.h, 794
FindServiceEventURLPath
service_table.h, 794
FindServiceId
service_table.h, 794
foreign_sockaddr
mserv_request_t, 255
Network Socket Library, 160
format_uuid_v1
uuid.c, 930
format_uuid_v3
uuid.c, 930
free_action_arg
UPnP API, 36
free_advertise_arg
UPnP API, 36
free_client_subscription
client_table.h, 732
free_func
LINKEDLIST, 251
free_function
LinkedList.h, 872
free_notify_struct
gena_device.c, 669
free_routine
ThreadPool.h, 902
free_ssdp_event_handler_data
SSDP Library, 168
free_subscribe_arg
gena_ctrlpt.c, 662
free_URL_list
uri.c, 721
uri.h, 834
freeClientSubList
client_table.h, 732
FreeExtraHTTPHeaders
webserver.c, 708
FreeHandle
UPnP API, 37
FREELIST, 235
FreeList
FreeList.h, 867
FreeList.h
FreeList, 867
FreeListAlloc, 867
FreeListDestroy, 868
FreeListFree, 868
FreeListInit, 868
FreeListNode, 867
FreeListAlloc
FreeList.h, 867
FreeListDestroy
FreeList.h, 868
FreeListFree
FreeList.h, 868
FreeListInit
FreeList.h, 868
FREELISTNODE, 236
FreeListNode
FreeList.h, 867
freeNodeList
LINKEDLIST, 251
freeService
service_table.h, 795

freeServiceList
service_table.h, 795
freeServiceTable
service_table.h, 795
freeSubscription
service_table.h, 796
freeSubscriptionList
service_table.h, 796
FreeThreadPoolJob
ThreadPool.c, 884
FreeTimerEvent
TimerThread.c, 918

g_log_level
upnpdebug.c, 593
g_maxContentLength
UPnP API, 91
g_UpnpSdkEQMaxAge
UPnP API, 91
g_UpnpSdkEQMaxLen
UPnP API, 92
gAliasDoc
webserver.c, 717
gAllowLiteralHostRedirection
UPnP API, 92
gDocumentRootDir
UPnP API, 92
gena.h
error_respond, 740
genaCallback, 740
genaInitNotify, 741
genaInitNotifyExt, 742
genaNotifyAll, 742
genaNotifyAllExt, 743
genaRenewSubscription, 743
genaSubscribe, 744
genaUnregisterClient, 745
genaUnregisterDevice, 745
genaUnSubscribe, 746
notify_thread_struct, 740
SubscribeLock, 739
SubscribeUnlock, 739
XML_VERSION, 740
gena_ctrlpt.c
free_subscribe_arg, 662
gena_process_notification_event, 662
gena_subscribe, 663
gena_unsubscribe, 663
GenaAutoRenewSubscription, 664
genaRenewSubscription, 664
genaSubscribe, 665
genaUnregisterClient, 665
genaUnSubscribe, 666
ScheduleGenaAutoRenew, 666
gena_ctrlpt.h
gena_process_notification_event, 749
gena_device.c
AllocGenaHeaders, 668
create_url_list, 669
free_notify_struct, 669
gena_process_subscription_renewal_request, 670
gena_process_subscription_request, 670
gena_process_unsubscribe_request, 670
gena_validate_delivery_urls, 671
genaInitNotify, 671
genaInitNotifyExt, 672
genaNotify, 672
genaNotifyAll, 673
genaNotifyAllExt, 674
genaNotifyThread, 674
genaUnregisterDevice, 675
GeneratePropertySet, 675
notify_send_and_recv, 676
respond_ok, 676
gena_device.h
gena_process_subscription_renewal_request, 751
gena_process_subscription_request, 752
gena_process_unsubscribe_request, 752
gena_process_notification_event
gena_ctrlpt.c, 662
gena_ctrlpt.h, 749
gena_process_subscription_renewal_request
gena_device.c, 670
gena_device.h, 751
gena_process_subscription_request
gena_device.c, 670
gena_device.h, 752
gena_process_unsubscribe_request
gena_device.c, 670
gena_device.h, 752
gena_subscribe
gena_ctrlpt.c, 663
gena_unsubscribe
gena_ctrlpt.c, 663
gena_validate_delivery_urls
gena_device.c, 671
GenaAutoRenewSubscription
gena_ctrlpt.c, 664
genaCallback
gena.h, 740
genaInitNotify
gena.h, 741
gena_device.c, 671
genaInitNotifyExt
gena.h, 742
gena_device.c, 672
genaNotify
gena_device.c, 672
genaNotifyAll
gena.h, 742
gena_device.c, 673
genaNotifyAllExt
gena.h, 743
gena_device.c, 674
genaNotifyThread
gena_device.c, 674
genaRenewSubscription

gena.h, 743
gena_ctrlpt.c, 664
genaSubscribe
gena.h, 744
gena_ctrlpt.c, 665
genaUnregisterClient
gena.h, 745
gena_ctrlpt.c, 665
genaUnregisterDevice
gena.h, 745
gena_device.c, 675
genaUnSubscribe
gena.h, 746
gena_ctrlpt.c, 666
GeneratePropertySet
gena_device.c, 675
GenlibClientSubscription
GenlibClientSubscription.h, 755
GenlibClientSubscription.c
GenlibClientSubscription_assign, 678
GenlibClientSubscription_clear_ActualSID, 678
GenlibClientSubscription_clear_EventURL, 679
GenlibClientSubscription_clear_SID, 679
GenlibClientSubscription_delete, 679
GenlibClientSubscription_dup, 679
GenlibClientSubscription_get_ActualSID, 679
GenlibClientSubscription_get_ActualSID_cstr, 680
GenlibClientSubscription_get_ActualSID_Length, 680
GenlibClientSubscription_get_EventURL, 680
GenlibClientSubscription_get_EventURL_cstr, 680
GenlibClientSubscription_get_EventURL_Length, 680
GenlibClientSubscription_get_Next, 681
GenlibClientSubscription_get_RenewEventId, 681
GenlibClientSubscription_get_SID, 681
GenlibClientSubscription_get_SID_cstr, 681
GenlibClientSubscription_get_SID_Length, 681
GenlibClientSubscription_new, 682
GenlibClientSubscription_set_ActualSID, 682
GenlibClientSubscription_set_EventURL, 682
GenlibClientSubscription_set_Next, 682
GenlibClientSubscription_set_RenewEventId, 682
GenlibClientSubscription_set_SID, 683
GenlibClientSubscription_strcpy_ActualSID, 683
GenlibClientSubscription_strcpy_EventURL, 683
GenlibClientSubscription_strcpy_SID, 683
GenlibClientSubscription_strncpy_ActualSID, 684
GenlibClientSubscription_strncpy_EventURL, 684
GenlibClientSubscription_strncpy_SID, 684
GenlibClientSubscription.h
GenlibClientSubscription, 755
GenlibClientSubscription_assign, 756
GenlibClientSubscription_clear_ActualSID, 756
GenlibClientSubscription_clear_EventURL, 756
GenlibClientSubscription_clear_SID, 756
GenlibClientSubscription_delete, 756
GenlibClientSubscription_dup, 756
GenlibClientSubscription_get_ActualSID, 757
GenlibClientSubscription_get_ActualSID_cstr, 757
GenlibClientSubscription_get_ActualSID_Length, 757
GenlibClientSubscription_get_EventURL, 757
GenlibClientSubscription_get_EventURL_cstr, 757
GenlibClientSubscription_get_EventURL_Length, 758
GenlibClientSubscription_get_Next, 758
GenlibClientSubscription_get_RenewEventId, 758
GenlibClientSubscription_get_SID, 758
GenlibClientSubscription_get_SID_cstr, 758
GenlibClientSubscription_get_SID_Length, 759
GenlibClientSubscription_new, 759
GenlibClientSubscription_set_ActualSID, 759
GenlibClientSubscription_set_EventURL, 759
GenlibClientSubscription_set_Next, 759
GenlibClientSubscription_set_RenewEventId, 760
GenlibClientSubscription_set_SID, 760
GenlibClientSubscription_strcpy_ActualSID, 760
GenlibClientSubscription_strcpy_EventURL, 760
GenlibClientSubscription_strcpy_SID, 760
GenlibClientSubscription_strncpy_ActualSID, 761
GenlibClientSubscription_strncpy_EventURL, 761
GenlibClientSubscription_strncpy_SID, 761
GenlibClientSubscription_assign
GenlibClientSubscription.c, 678
GenlibClientSubscription.h, 756
GenlibClientSubscription_clear_ActualSID
GenlibClientSubscription.c, 678
GenlibClientSubscription.h, 756
GenlibClientSubscription_clear_EventURL
GenlibClientSubscription.c, 679
GenlibClientSubscription.h, 756
GenlibClientSubscription_clear_SID
GenlibClientSubscription.c, 679
GenlibClientSubscription.h, 756
GenlibClientSubscription_delete
GenlibClientSubscription.c, 679
GenlibClientSubscription.h, 756
GenlibClientSubscription_dup
GenlibClientSubscription.c, 679
GenlibClientSubscription.h, 756
GenlibClientSubscription_get_ActualSID
GenlibClientSubscription.c, 679
GenlibClientSubscription.h, 757
GenlibClientSubscription_get_ActualSID_cstr
GenlibClientSubscription.c, 680
GenlibClientSubscription.h, 757
GenlibClientSubscription_get_ActualSID_Length
GenlibClientSubscription.c, 680
GenlibClientSubscription.h, 757
GenlibClientSubscription_get_EventURL
GenlibClientSubscription.c, 680
GenlibClientSubscription.h, 757
GenlibClientSubscription_get_EventURL_cstr
GenlibClientSubscription.c, 680
GenlibClientSubscription.h, 757

GenlibClientSubscription_get_EventURL_Length
GenlibClientSubscription.c, 680
GenlibClientSubscription.h, 758
GenlibClientSubscription_get_Next
GenlibClientSubscription.c, 681
GenlibClientSubscription.h, 758
GenlibClientSubscription_get_RenewEventId
GenlibClientSubscription.c, 681
GenlibClientSubscription.h, 758
GenlibClientSubscription_get_SID
GenlibClientSubscription.c, 681
GenlibClientSubscription.h, 758
GenlibClientSubscription_get_SID_cstr
GenlibClientSubscription.c, 681
GenlibClientSubscription.h, 758
GenlibClientSubscription_get_SID_Length
GenlibClientSubscription.c, 681
GenlibClientSubscription.h, 759
GenlibClientSubscription_new
GenlibClientSubscription.c, 682
GenlibClientSubscription.h, 759
GenlibClientSubscription_set_ActualSID
GenlibClientSubscription.c, 682
GenlibClientSubscription.h, 759
GenlibClientSubscription_set_EventURL
GenlibClientSubscription.c, 682
GenlibClientSubscription.h, 759
GenlibClientSubscription_set_Next
GenlibClientSubscription.c, 682
GenlibClientSubscription.h, 759
GenlibClientSubscription_set_RenewEventId
GenlibClientSubscription.c, 682
GenlibClientSubscription.h, 760
GenlibClientSubscription_set_SID
GenlibClientSubscription.c, 683
GenlibClientSubscription.h, 760
GenlibClientSubscription_strcpy_ActualSID
GenlibClientSubscription.c, 683
GenlibClientSubscription.h, 760
GenlibClientSubscription_strcpy_EventURL
GenlibClientSubscription.c, 683
GenlibClientSubscription.h, 760
GenlibClientSubscription_strcpy_SID
GenlibClientSubscription.c, 683
GenlibClientSubscription.h, 760
GenlibClientSubscription_strncpy_ActualSID
GenlibClientSubscription.c, 684
GenlibClientSubscription.h, 761
GenlibClientSubscription_strncpy_EventURL
GenlibClientSubscription.c, 684
GenlibClientSubscription.h, 761
GenlibClientSubscription_strncpy_SID
GenlibClientSubscription.c, 684
GenlibClientSubscription.h, 761
get_alias
webserver.c, 708
get_content_type
webserver.c, 709
get_current_time
uuid.c, 930
get_dev_service
soap_device.c, 847
get_file_info
webserver.c, 709
get_ieee_node_identifier
sysdep.c, 928
sysdep.h, 820
get_info
VirtualDirCallbacks, 297
get_miniserver_stopsock
miniserver.c, 687
get_mpost_acton_hdrval
soap_device.c, 848
get_port
miniserver.c, 687
get_ssdp_sockets
SSDP Library, 169
get_system_time
sysdep.c, 928
sysdep.h, 820
GetCallBackFn
UPnP API, 37
GetClientHandleInfo
UPnP API, 37
GetClientSubActualSID
client_table.h, 733
GetClientSubClientSID
client_table.h, 733
GetDescDocumentAndURL
UPnP API, 38
GetDeviceHandleInfo
UPnP API, 38
GetDeviceHandleInfoForPath
UPnP API, 39
getElementValue
service_table.h, 796
GetFirstSubscription
service_table.h, 796
GetFreeHandle
UPnP API, 39
GetHandleInfo
UPnP API, 39
GetNextRange
webserver.c, 710
GetNextSubscription
service_table.h, 797
getServiceTable
service_table.h, 797
getSubElement
service_table.h, 798
GetSubscriptionSID
service_table.h, 798
gIF_INDEX
UPnP API, 92
gIF_IPV4
UPnP API, 92

gIF_IPV4_NETMASK
UPnP API, 92
gIF_IPV6
UPnP API, 93
gIF_IPV6_PREFIX_LENGTH
UPnP API, 93
gIF_IPV6_ULA_GUA
UPnP API, 93
gIF_IPV6_ULA_GUA_PREFIX_LENGTH
UPnP API, 93
gIF_NAME
UPnP API, 93
glob_alias_init
webserver.c, 710
GlobalClientSubscribeMutex
UPnP API, 93
GlobalDebugMutex
upnpdebug.c, 593
GlobalDeviceList
Sample Code, 150, 151
GlobalHndRWLock
UPnP API, 94
gMediaTypeList
webserver.c, 717
gMediaTypes
webserver.c, 717
gMiniServerThreadPool
UPnP API, 94
gMServState
miniserver.c, 689
gPrintFun
Sample Code, 151
gRecvThreadPool
UPnP API, 94
gSDKInitMutex
UPnP API, 94
gSendThreadPool
UPnP API, 94
gStateUpdateFun
Sample Code, 151
gTimerThread
UPnP API, 94
gUpnpSdkNLSuuid
upnpapi.h, 825
gUUIDMutex
UPnP API, 95
gWebCallback_HostValidate
UPnP API, 95
gWebCallback_HostValidateCookie
UPnP API, 95
gWebserverCorsString
UPnP API, 95

Handle_Info, 237
aliasInstalled, 238
Callback, 238
ClientSubList, 238
Cookie, 238
DescDocument, 238
DescURL, 238
DescXML, 239
DeviceAf, 239
DeviceList, 239
HType, 239
LowerDescURL, 239
MaxSubscriptions, 239
MaxSubscriptionTimeOut, 240
ServiceList, 240
ServiceTable, 240
SsdpSearchList, 240
handle_invoke_action
soap_device.c, 848
handle_query_variable
soap_device.c, 849
HandleReadLock
upnpapi.h, 824
HandleTable
UPnP API, 95
HandleUnlock
upnpapi.h, 824
HandleWriteLock
upnpapi.h, 824
has_xml_content_type
parsetools.c, 703
parsetools.h, 790
head
LINKEDLIST, 251
HEADER_LENGTH
upnptools.c, 659
headers
http_message_t, 244
highJobQ
THREADPOOL, 278
HOSTPORT, 241
text, 241
hostport_type
uri.h, 833
http_CloseHttpConnection
httpreadwrite.c, 692
http_Connect
httpreadwrite.c, 693
http_EndHttpRequest
httpreadwrite.c, 693
http_error_code
http_parser_t, 246
http_FixStrUrl
httpreadwrite.c, 694
http_FixUrl
httpreadwrite.c, 694
http_GetHttpResponse
httpreadwrite.c, 695
http_header_t, 242
name, 242
name_id, 242
value, 242
http_MakeHttpRequest
httpreadwrite.c, 696

http_MakeMessage
httpreadwrite.c, 697
http_message_t, 243
amount_discarded, 243
entity, 243
headers, 244
is_request, 244
method, 244
msg, 244
request_method, 244
status_code, 244
status_msg, 245
uri, 245
urlbuf, 245
http_OpenHttpConnection
httpreadwrite.c, 698
http_parser_t, 246
entity_start_position, 246
http_error_code, 246
valid_ssdp_notify_hack, 247
http_ReadHttpResponse
httpreadwrite.c, 698
http_RecvMessage
httpreadwrite.c, 699
http_RecvPostMessage
webserver.c, 710
http_SendMessage
httpreadwrite.c, 700
http_WriteHttpRequest
httpreadwrite.c, 700
HTTPCONNECTIONHANDLE, 247
httpparser.h
PARSE_CONTINUE_1, 765
PARSE_FAILURE, 765
PARSE_INCOMPLETE, 765
PARSE_INCOMPLETE_ENTITY, 765
PARSE_NO_MATCH, 765
PARSE_OK, 765
parse_status_t, 765
PARSE_SUCCESS, 765
print_http_headers, 765
httpreadwrite.c
Check_Connect_And_Wait_Connection, 692
http_CloseHttpConnection, 692
http_Connect, 693
http_EndHttpRequest, 693
http_FixStrUrl, 694
http_FixUrl, 694
http_GetHttpResponse, 695
http_MakeHttpRequest, 696
http_MakeMessage, 697
http_OpenHttpConnection, 698
http_ReadHttpResponse, 698
http_RecvMessage, 699
http_SendMessage, 700
http_WriteHttpRequest, 700
ReadResponseLineAndHeaders, 701
HType
Handle_Info, 239

IN6_IS_ADDR_GLOBAL
UPnP API, 26
IN6_IS_ADDR_ULA
UPnP API, 26
IncrementBrightness
Sample Code, 125
IncrementColor
Sample Code, 125
IncrementContrast
Sample Code, 126
IncrementVolume
Sample Code, 126
INVALID_JOB_ID
ThreadPool.h, 902
INVALID_POLICY
ThreadPool.h, 902
INVALID_SOCKET
Network Socket Library, 155
is_escaped
uri.c, 722
is_mark
uri.c, 722
is_request
http_message_t, 244
is_reserved
uri.c, 723
is_unreserved
uri.c, 723
is_valid_alias
webserver.c, 711
isDuplicateAttribute
ixmlparser.c, 351
isFileInVirtualDir
webserver.c, 712
isTopLevelElement
ixmlparser.c, 351
isUrlV6UlaGua
SSDP Library, 169
ixml.c
copy_with_escape, 338
ixmlDomTreetoString, 338
ixmlPrintDomTree, 339
ixmlPrintDomTreeRecursive, 340
IXML_Attr
XML API, 183
IXML_CDATASection
XML API, 183
IXML_Document
XML API, 184
IXML_Element
XML API, 184
IXML_ERRORCODE
XML API, 185
ixml_membuf, 248
ixml_membuf_append
ixmlmembuf.c, 343
ixmlmembuf.h, 323

ixml_membuf_append_str
ixmlmembuf.c, 343
ixmlmembuf.h, 323
ixml_membuf_assign
ixmlmembuf.c, 344
ixmlmembuf.h, 324
ixml_membuf_assign_str
ixmlmembuf.c, 345
ixmlmembuf.h, 325
ixml_membuf_destroy
ixmlmembuf.c, 345
ixmlmembuf.h, 325
ixml_membuf_init
ixmlmembuf.c, 345
ixmlmembuf.h, 325
ixml_membuf_insert
ixmlmembuf.c, 346
ixmlmembuf.h, 326
ixml_membuf_set_size
ixmlmembuf.c, 346
IXML_NamedNodeMap
XML API, 184
IXML_Node
XML API, 184
IXML_NODE_TYPE
XML API, 185
IXML_NodeList
XML API, 184
IXML_Text
XML API, 184
ixmlAttr_free
XML API, 185
ixmlCDATASection_free
XML API, 185
ixmlCDATASection_init
XML API, 185
ixmlCloneDOMString
XML API, 186
ixmldebug.c
IxmlPrintf, 341
printNodes, 342
ixmldebug.h
IxmlPrintf, 316
printNodes, 316
ixmlDocument_createAttribute
XML API, 186
ixmlDocument_createAttributeEx
XML API, 187
ixmlDocument_createAttributeNS
XML API, 187
ixmlDocument_createAttributeNSEx
XML API, 188
ixmlDocument_createCDATASection
XML API, 188
ixmlDocument_createCDATASectionEx
XML API, 189
ixmlDocument_createDocument
XML API, 189
ixmlDocument_createDocumentEx
XML API, 189
ixmlDocument_createElement
XML API, 190
ixmlDocument_createElementEx
XML API, 190
ixmlDocument_createElementNS
XML API, 191
ixmlDocument_createElementNSEx
XML API, 191
ixmlDocument_createTextNode
XML API, 192
ixmlDocument_createTextNodeEx
XML API, 192
ixmlDocument_free
XML API, 193
ixmlDocument_getElementById
XML API, 193
ixmlDocument_getElementsByTagName
XML API, 193
ixmlDocument_getElementsByTagNameNS
XML API, 194
ixmlDocument_importNode
XML API, 194
ixmlDocument_init
XML API, 195
ixmlDocument_setOwnerDocument
document.c, 319
ixmlDocumenttoString
XML API, 195
ixmlDomTreetoString
ixml.c, 338
ixmlElement_findAttributeNode
element.c, 321
ixmlElement_free
XML API, 196
ixmlElement_getAttribute
XML API, 196
ixmlElement_getAttributeNode
XML API, 196
ixmlElement_getAttributeNodeNS
XML API, 197
ixmlElement_getAttributeNS
XML API, 197
ixmlElement_getElementsByTagName
XML API, 198
ixmlElement_getElementsByTagNameNS
XML API, 198
ixmlElement_getTagName
XML API, 198
ixmlElement_hasAttribute
XML API, 199
ixmlElement_hasAttributeNS
XML API, 199
ixmlElement_init
XML API, 199
ixmlElement_removeAttribute
XML API, 200

ixmlElement_removeAttributeNode
XML API, 200
ixmlElement_removeAttributeNS
XML API, 201
ixmlElement_setAttribute
XML API, 201
ixmlElement_setAttributeNode
XML API, 202
ixmlElement_setAttributeNodeNS
XML API, 202
ixmlElement_setAttributeNS
XML API, 203
ixmlElement_setTagName
element.c, 321
ixmlparser.h, 330
ixmlFreeDOMString
XML API, 204
ixmlLoadDocument
XML API, 204
ixmlLoadDocumentEx
XML API, 204
ixmlmembuf.c
ixml_membuf_append, 343
ixml_membuf_append_str, 343
ixml_membuf_assign, 344
ixml_membuf_assign_str, 345
ixml_membuf_destroy, 345
ixml_membuf_init, 345
ixml_membuf_insert, 346
ixml_membuf_set_size, 346
ixmlmembuf.h
ixml_membuf_append, 323
ixml_membuf_append_str, 323
ixml_membuf_assign, 324
ixml_membuf_assign_str, 325
ixml_membuf_destroy, 325
ixml_membuf_init, 325
ixml_membuf_insert, 326
ixmlNamedNodeMap_addToNamedNodeMap
ixmlparser.h, 330
namedNodeMap.c, 372
ixmlNamedNodeMap_free
XML API, 205
ixmlNamedNodeMap_getItemNumber
namedNodeMap.c, 372
ixmlNamedNodeMap_getLength
XML API, 205
ixmlNamedNodeMap_getNamedItem
XML API, 205
ixmlNamedNodeMap_getNamedItemNS
XML API, 206
ixmlNamedNodeMap_init
ixmlparser.h, 330
namedNodeMap.c, 372
ixmlNamedNodeMap_item
XML API, 206
ixmlNamedNodeMap_removeNamedItem
XML API, 206
ixmlNamedNodeMap_removeNamedItemNS
XML API, 207
ixmlNamedNodeMap_setNamedItem
XML API, 207
ixmlNamedNodeMap_setNamedItemNS
XML API, 208
ixmlNode_allowChildren
node.c, 375
ixmlNode_appendChild
XML API, 208
ixmlNode_cloneAttr
node.c, 376
ixmlNode_cloneAttrDirect
node.c, 376
ixmlNode_cloneCDATASect
node.c, 377
ixmlNode_cloneElement
node.c, 377
ixmlNode_cloneNode
XML API, 209
ixmlNode_cloneNodeTree
node.c, 377
ixmlNode_cloneNodeTreeRecursive
node.c, 378
ixmlNode_cloneTextNode
node.c, 378
ixmlNode_compare
ixmlparser.h, 331
node.c, 379
ixmlNode_free
XML API, 209
ixmlNode_freeSingleNode
node.c, 379
ixmlNode_getAttributes
XML API, 209
ixmlNode_getChildNodes
XML API, 210
ixmlNode_getElementsByTagName
ixmlparser.h, 331
node.c, 379
ixmlNode_getElementsByTagNameNS
ixmlparser.h, 331
node.c, 380
ixmlNode_getElementsByTagNameNSRecursive
node.c, 380
ixmlNode_getElementsByTagNameRecursive
node.c, 380
ixmlNode_getFirstChild
XML API, 210
ixmlNode_getLastChild
XML API, 211
ixmlNode_getLocalName
XML API, 211
ixmlNode_getNamespaceURI
XML API, 211
ixmlNode_getNextSibling
XML API, 212
ixmlNode_getNodeName

XML API, 212
ixmlNode_getNodeType
XML API, 213
ixmlNode_getNodeValue
XML API, 213
ixmlNode_getOwnerDocument
XML API, 214
ixmlNode_getParentNode
XML API, 214
ixmlNode_getPrefix
XML API, 214
ixmlNode_getPreviousSibling
XML API, 215
ixmlNode_hasAttributes
XML API, 215
ixmlNode_hasChildNodes
XML API, 215
ixmlNode_init
ixmlparser.h, 332
node.c, 381
ixmlNode_insertBefore
XML API, 216
ixmlNode_isAncestor
node.c, 381
ixmlNode_isParent
node.c, 382
ixmlNode_newDoc
node.c, 382
ixmlNode_removeChild
XML API, 216
ixmlNode_replaceChild
XML API, 217
ixmlNode_setLocalName
node.c, 382
ixmlNode_setNamespaceURI
node.c, 383
ixmlNode_setNodeName
ixmlparser.h, 332
node.c, 383
ixmlNode_setNodeProperties
ixmlparser.h, 332
node.c, 383
ixmlNode_setNodeValue
XML API, 218
ixmlNode_setPrefix
node.c, 384
ixmlNode_setSiblingNodesParent
node.c, 384
ixmlNodeList_addToNodeList
ixmlparser.h, 333
nodeList.c, 385
ixmlNodeList_free
XML API, 218
ixmlNodeList_init
ixmlparser.h, 333
nodeList.c, 386
ixmlNodeList_item
XML API, 219
ixmlNodeList_length
XML API, 219
ixmlNodetoString
XML API, 219
ixmlParseBuffer
XML API, 220
ixmlParseBufferEx
XML API, 220
ixmlparser.c
isDuplicateAttribute, 351
isTopLevelElement, 351
Letter, 370
LETTERTABLESIZE, 351
NameChar, 371
NAMECHARTABLESIZE, 351
Parser_addNamespace, 352
Parser_appendTokBufChar, 352
Parser_appendTokBufStr, 352
Parser_clearTokenBuf, 353
Parser_copyToken, 353
Parser_ElementPrefixDefined, 353
Parser_eTagVerification, 354
Parser_free, 354
Parser_freeElementStackItem, 354
Parser_freeNodeContent, 356
Parser_freeNsURI, 356
Parser_getChar, 356
Parser_getNameSpace, 356
Parser_getNextNode, 357
Parser_getNextToken, 357
Parser_hasDefaultNamespace, 358
Parser_init, 358
Parser_intToUTF8, 358
Parser_isCharInTable, 359
Parser_isNameChar, 359
Parser_isValidEndElement, 359
Parser_isValidXmlName, 360
Parser_isXmlChar, 360
Parser_LoadDocument, 360
Parser_parseDocument, 361
Parser_popElement, 361
Parser_processAttribute, 361
Parser_processAttributeName, 362
Parser_processCDSect, 362
Parser_processContent, 363
Parser_processElementName, 363
Parser_processETag, 364
Parser_processSTag, 364
Parser_pushElement, 364
Parser_readFileOrBuffer, 365
Parser_setElementNamespace, 365
Parser_setErrorChar, 365
Parser_setLastElem, 366
Parser_setNodePrefixAndLocalName, 366
Parser_skipBom, 366
Parser_skipComment, 367
Parser_skipDocType, 367
Parser_skipMisc, 367

Parser_skipPI, 368
Parser_skipProlog, 368
Parser_skipString, 368
Parser_skipWhiteSpaces, 369
Parser_skipXMLDecl, 369
Parser_UTF8ToInt, 369
Parser_xmlNamespace, 370
safe_strdup, 370
ixmlparser.h
ixmlElement_setTagName, 330
ixmlNamedNodeMap_addToNamedNodeMap, 330
ixmlNamedNodeMap_init, 330
ixmlNode_compare, 331
ixmlNode_getElementsByTagName, 331
ixmlNode_getElementsByTagNameNS, 331
ixmlNode_init, 332
ixmlNode_setNodeName, 332
ixmlNode_setNodeProperties, 332
ixmlNodeList_addToNodeList, 333
ixmlNodeList_init, 333
Parser_freeNodeContent, 333
Parser_isValidXmlName, 334
Parser_LoadDocument, 334
Parser_setErrorChar, 334
Parser_setNodePrefixAndLocalName, 335
ixmlPrintDocument
XML API, 221
ixmlPrintDomTree
ixml.c, 339
ixmlPrintDomTreeRecursive
ixml.c, 340
IxmlPrintf
ixmldebug.c, 341
ixmldebug.h, 316
ixmlPrintNode
XML API, 221
ixmlRelaxParser
XML API, 222

job_arg, 249
jobFreeList
THREADPOOL, 278
JOBFREELISTSIZE
ThreadPool.h, 902
jobsPerThread
THREADPOOLATTR, 281

last_modified
xml_alias_t, 299
lastJobId
THREADPOOL, 278
length
membuffer, 254
memptr, 255
Letter
ixmlparser.c, 370
LETTERTABLESIZE
ixmlparser.c, 351
linecopy
upnputil.h, 829
util.c, 730
linecopylen
upnputil.h, 829
util.c, 730
LINKEDLIST, 250
cmp_func, 251
free_func, 251
freeNodeList, 251
head, 251
size, 251
tail, 251
LinkedList
LinkedList.h, 872
LinkedList.h
cmp_routine, 872
free_function, 872
LinkedList, 872
ListAddAfter, 872
ListAddBefore, 873
ListAddHead, 873
ListAddTail, 874
ListDelNode, 874
ListDestroy, 874
ListFind, 875
ListHead, 875
ListInit, 876
ListNext, 876
ListNode, 872
ListPrev, 877
ListSize, 877
ListTail, 877
linux_print
Sample Code, 127
ListAddAfter
LinkedList.h, 872
ListAddBefore
LinkedList.h, 873
ListAddHead
LinkedList.h, 873
ListAddTail
LinkedList.h, 874
ListDelNode
LinkedList.h, 874
ListDestroy
LinkedList.h, 874
ListFind
LinkedList.h, 875
ListHead
LinkedList.h, 875
ListInit
LinkedList.h, 876
ListNext
LinkedList.h, 876
LISTNODE, 252
ListNode
LinkedList.h, 872
ListPrev

LinkedList.h, 877
ListSize
LinkedList.h, 877
ListTail
LinkedList.h, 877
LOCAL_PORT_V4
UPnP API, 95
LOCAL_PORT_V6
UPnP API, 96
LOCAL_PORT_V6_ULA_GUA
UPnP API, 96
LowerDescURL
Handle_Info, 239
lowJobQ
THREADPOOL, 279

m_length
The UpnpString Class, 105
m_string
The UpnpString Class, 105
makeAction
upnptools.c, 660
MAX_BRIGHTNESS
Sample Code, 118
MAX_CHANNEL
Sample Code, 118
MAX_COLOR
Sample Code, 119
MAX_CONTRAST
Sample Code, 119
MAX_TINT
Sample Code, 119
MAX_VOLUME
Sample Code, 119
maxIdleTime
THREADPOOLATTR, 281
maxJobsTotal
THREADPOOLATTR, 281
MaxSubscriptions
Handle_Info, 239
MaxSubscriptionTimeOut
Handle_Info, 240
maxThreads
THREADPOOLATTR, 281
MD5Context, 253
media_list_init
webserver.c, 712
medJobQ
THREADPOOL, 279
MEMBUF_DEF_SIZE_INC
membuffer.h, 777
membuffer, 253
buf, 253
capacity, 253
length, 254
size_inc, 254
membuffer.h
MEMBUF_DEF_SIZE_INC, 777
membuffer_append, 777
membuffer_append_str, 778
membuffer_assign, 778
membuffer_assign_str, 779
membuffer_attach, 779
membuffer_delete, 779
membuffer_destroy, 780
membuffer_detach, 780
membuffer_init, 780
membuffer_insert, 780
membuffer_set_size, 781
memptr_cmp, 781
memptr_cmp_nocase, 782
str_alloc, 782
membuffer_append
membuffer.h, 777
membuffer_append_str
membuffer.h, 778
membuffer_assign
membuffer.h, 778
membuffer_assign_str
membuffer.h, 779
membuffer_attach
membuffer.h, 779
membuffer_delete
membuffer.h, 779
membuffer_destroy
membuffer.h, 780
membuffer_detach
membuffer.h, 780
membuffer_init
membuffer.h, 780
membuffer_insert
membuffer.h, 780
membuffer_set_size
membuffer.h, 781
memptr, 254
buf, 255
length, 255
memptr_cmp
membuffer.h, 781
memptr_cmp_nocase
membuffer.h, 782
method
http_message_t, 244
miniserver.c
APPLICATION_LISTENING_PORT, 686
get_miniserver_stopsock, 687
get_port, 687
gMServState, 689
MiniServerState, 686
miniStopSockPort, 689
MSERV_IDLE, 687
MSERV_RUNNING, 687
MSERV_STOPPING, 687
RunMiniServer, 688
StartMiniServer, 688
StopMiniServer, 689
miniserver.h

MiniServerCallback, 786
SetGenaCallback, 786
SetHTTPGetCallback, 786
SetSoapCallback, 786
StartMiniServer, 786
StopMiniServer, 787
MiniServerCallback
miniserver.h, 786
miniServerSock4
MServerSockArray, 256
miniServerSock6
MServerSockArray, 256
miniServerSock6UlaGua
MServerSockArray, 256
MiniServerState
miniserver.c, 686
miniServerStopSock
MServerSockArray, 256
miniStopSockPort
miniserver.c, 689
minThreads
THREADPOOLATTR, 282
MSERV_IDLE
miniserver.c, 687
mserv_request_t, 255
connfd, 255
foreign_sockaddr, 255
MSERV_RUNNING
miniserver.c, 687
MSERV_STOPPING
miniserver.c, 687
MServerSockArray, 256
miniServerSock4, 256
miniServerSock6, 256
miniServerSock6UlaGua, 256
miniServerStopSock, 256
ssdpReqSock4, 257
ssdpReqSock6, 257
ssdpSock4, 257
ssdpSock6, 257
ssdpSock6UlaGua, 257
msg
http_message_t, 244
scanner_t, 267
mutex
THREADPOOL, 279

name
http_header_t, 242
xml_alias_t, 299
name_id
http_header_t, 242
NameChar
ixmlparser.c, 371
NAMECHARTABLESIZE
ixmlparser.c, 351
namecopy
upnputil.h, 829
util.c, 731
namedNodeMap.c
ixmlNamedNodeMap_addToNamedNodeMap, 372
ixmlNamedNodeMap_getItemNumber, 372
ixmlNamedNodeMap_init, 372
Network Socket Library, 154
foreign_sockaddr, 160
INVALID_SOCKET, 155
sock_close, 155
sock_destroy, 156
sock_init, 156
sock_init_with_ip, 157
sock_make_blocking, 157
sock_make_no_blocking, 158
sock_read, 158
sock_read_write, 158
sock_write, 159
SOCKET, 155
socket, 160
SOCKET_ERROR, 155
UpnpCloseSocket, 155
NewRequestHandler
SSDP Library, 169
NO_ERROR_FOUND
SSDP Library, 164
node
_uuid_upnp, 232
uuid_state, 295
node.c
ixmlNode_allowChildren, 375
ixmlNode_cloneAttr, 376
ixmlNode_cloneAttrDirect, 376
ixmlNode_cloneCDATASect, 377
ixmlNode_cloneElement, 377
ixmlNode_cloneNodeTree, 377
ixmlNode_cloneNodeTreeRecursive, 378
ixmlNode_cloneTextNode, 378
ixmlNode_compare, 379
ixmlNode_freeSingleNode, 379
ixmlNode_getElementsByTagName, 379
ixmlNode_getElementsByTagNameNS, 380
ixmlNode_getElementsByTagNameNSRecursive, 380
ixmlNode_getElementsByTagNameRecursive, 380
ixmlNode_init, 381
ixmlNode_isAncestor, 381
ixmlNode_isParent, 382
ixmlNode_newDoc, 382
ixmlNode_setLocalName, 382
ixmlNode_setNamespaceURI, 383
ixmlNode_setNodeName, 383
ixmlNode_setNodeProperties, 383
ixmlNode_setPrefix, 384
ixmlNode_setSiblingNodesParent, 384
nodeList.c
ixmlNodeList_addToNodeList, 385
ixmlNodeList_init, 386
notify_send_and_recv
gena_device.c, 676

NOTIFY_THREAD_STRUCT, 258
notify_thread_struct
    gena.h, 740

open
    VirtualDirCallbacks, 297
Optional Tool API, 105
    UpnpAddToAction, 106
    UpnpAddToActionResponse, 107
    UpnpAddToPropertySet, 107
    UpnpCreatePropertySet, 108
    UpnpGetErrorMessage, 108
    UpnpMakeAction, 109
    UpnpMakeActionResponse, 109
    UpnpResolveURL, 110
    UpnpResolveURL2, 111
    UpnpSetMaxJobsTotal, 111

PARSE_CONTINUE_1
    httpparser.h, 765
PARSE_FAILURE
    httpparser.h, 765
parse_hostport
    uri.c, 723
PARSE_INCOMPLETE
    httpparser.h, 765
PARSE_INCOMPLETE_ENTITY
    httpparser.h, 765
PARSE_NO_MATCH
    httpparser.h, 765
PARSE_OK
    httpparser.h, 765
parse_scheme
    uri.c, 725
parse_status_t
    httpparser.h, 765
PARSE_SUCCESS
    httpparser.h, 765
parse_token
    uri.h, 834
parse_uri
    uri.c, 725
    uri.h, 834
parse_uric
    uri.c, 726
Parser_addNamespace
    ixmlparser.c, 352
Parser_appendTokBufChar
    ixmlparser.c, 352
Parser_appendTokBufStr
    ixmlparser.c, 352
Parser_clearTokenBuf
    ixmlparser.c, 353
Parser_copyToken
    ixmlparser.c, 353
Parser_ElementPrefixDefined
    ixmlparser.c, 353
Parser_eTagVerification
    ixmlparser.c, 354
Parser_free
    ixmlparser.c, 354
Parser_freeElementStackItem
    ixmlparser.c, 354
Parser_freeNodeContent
    ixmlparser.c, 356
    ixmlparser.h, 333
Parser_freeNsURI
    ixmlparser.c, 356
Parser_getChar
    ixmlparser.c, 356
Parser_getNameSpace
    ixmlparser.c, 356
Parser_getNextNode
    ixmlparser.c, 357
Parser_getNextToken
    ixmlparser.c, 357
Parser_hasDefaultNamespace
    ixmlparser.c, 358
Parser_init
    ixmlparser.c, 358
Parser_intToUTF8
    ixmlparser.c, 358
Parser_isCharInTable
    ixmlparser.c, 359
Parser_isNameChar
    ixmlparser.c, 359
Parser_isValidEndElement
    ixmlparser.c, 359
Parser_isValidXmlName
    ixmlparser.c, 360
    ixmlparser.h, 334
Parser_isXmlChar
    ixmlparser.c, 360
Parser_LoadDocument
    ixmlparser.c, 360
    ixmlparser.h, 334
Parser_parseDocument
    ixmlparser.c, 361
Parser_popElement
    ixmlparser.c, 361
Parser_processAttribute
    ixmlparser.c, 361
Parser_processAttributeName
    ixmlparser.c, 362
Parser_processCDSect
    ixmlparser.c, 362
Parser_processContent
    ixmlparser.c, 363
Parser_processElementName
    ixmlparser.c, 363
Parser_processETag
    ixmlparser.c, 364
Parser_processSTag
    ixmlparser.c, 364
Parser_pushElement
    ixmlparser.c, 364
Parser_readFileOrBuffer

ixmlparser.c, 365
Parser_setElementNamespace
ixmlparser.c, 365
Parser_setErrorChar
ixmlparser.c, 365
ixmlparser.h, 334
Parser_setLastElem
ixmlparser.c, 366
Parser_setNodePrefixAndLocalName
ixmlparser.c, 366
ixmlparser.h, 335
Parser_skipBom
ixmlparser.c, 366
Parser_skipComment
ixmlparser.c, 367
Parser_skipDocType
ixmlparser.c, 367
Parser_skipMisc
ixmlparser.c, 367
Parser_skipPI
ixmlparser.c, 368
Parser_skipProlog
ixmlparser.c, 368
Parser_skipString
ixmlparser.c, 368
Parser_skipWhiteSpaces
ixmlparser.c, 369
Parser_skipXMLDecl
ixmlparser.c, 369
Parser_UTF8ToInt
ixmlparser.c, 369
Parser_xmlNamespace
ixmlparser.c, 370
parsetools.c
has_xml_content_type, 703
parsetools.h
has_xml_content_type, 790
pendingWorkerThreadStart
THREADPOOL, 279
persistent
TIMEREVENT, 284
persistentJob
THREADPOOL, 279
persistentThreads
THREADPOOL, 279
POWER_ON
Sample Code, 120
print_http_headers
httpparser.h, 765
print_string
Sample Code, 123
print_token
uri.h, 835
print_uri
uri.h, 835
PrintHandleInfo
UPnP API, 40
printNodes
ixmldebug.c, 342
ixmldebug.h, 316
printService
service_table.h, 798
printServiceList
service_table.h, 799
printServiceTable
service_table.h, 799
PrintThreadPoolStats
UPnP API, 40
PRIzd
UpnpGlobal.h, 508
process_request
webserver.c, 712
pVirtualDirList
UPnP API, 96

rc
ErrorString, 235
rcError
ErrorString, 235
read
VirtualDirCallbacks, 297
read_state
uuid.c, 931
readFromSSDPSocket
SSDP Library, 170
ReadResponseLineAndHeaders
httpreadwrite.c, 701
ReadSendSize
SendInstruction, 268
RecvWriteSize
SendInstruction, 268
REL_SEC
TimerThread.h, 924
remove_dots
uri.c, 726
uri.h, 835
remove_escaped_chars
uri.c, 727
uri.h, 836
RemoveClientSubClientSID
client_table.h, 733
removeServiceTable
service_table.h, 799
RemoveSubscriptionSID
service_table.h, 800
replace_escaped
uri.c, 727
uri.h, 837
request_method
http_message_t, 244
RequestCookie
SendInstruction, 268
RESERVED
uri.h, 833
resolve_rel_url
uri.c, 728
uri.h, 837

resp_type
webserver.c, 706
respond_ok
gena_device.c, 676
RunMiniServer
miniserver.c, 688

s_Class, 258
s_GenlibClientSubscription, 259
s_Member, 259
s_SSDPResultData, 260
s_TestClass, 260
s_Type_Integer, 261
s_UpnpActionComplete, 261
s_UpnpActionRequest, 262
s_UpnpDiscovery, 263
s_UpnpEvent, 263
s_UpnpEventSubscribe, 264
s_UpnpExtraHeaders, 264
s_UpnpFileInfo, 265
s_UpnpStateVarComplete, 265
s_UpnpStateVarRequest, 266
s_UpnpSubscriptionRequest, 266
safe_strdup
ixmlparser.c, 370
Sample Code, 112
cmdloop_cmdlist, 149
cmdloop_tvcmds, 124
default_advr_expire, 149
default_timeout, 149
device_handle, 149
device_main, 125
DeviceListMutex, 150
display_mutex, 150
GlobalDeviceList, 150, 151
gPrintFun, 151
gStateUpdateFun, 151
IncrementBrightness, 125
IncrementColor, 125
IncrementContrast, 126
IncrementVolume, 126
linux_print, 127
MAX_BRIGHTNESS, 118
MAX_CHANNEL, 118
MAX_COLOR, 119
MAX_CONTRAST, 119
MAX_TINT, 119
MAX_VOLUME, 119
POWER_ON, 120
print_string, 123
SampleUtil_FindAndParseService, 127
SampleUtil_Finish, 127
SampleUtil_GetElementValue, 128
SampleUtil_GetFirstDocumentItem, 128
SampleUtil_GetFirstElementItem, 128
SampleUtil_GetFirstServiceList, 129
SampleUtil_Initialize, 129
SampleUtil_Print, 130
SampleUtil_PrintEvent, 130
SampleUtil_PrintEventType, 131
SampleUtil_RegisterUpdateFunction, 131
SampleUtil_StateUpdate, 132
SetActionTable, 132
SetServiceTable, 132
state_update, 124
TV_CONTROL_CHANNEL, 120
TV_CONTROL_POWER, 120
TV_CONTROL_VARCOUNT, 120
TV_CONTROL_VOLUME, 121
TV_MAX_VAL_LEN, 121
TV_MAXACTIONS, 121
TV_MAXVARS, 121
TV_PICTURE_BRIGHTNESS, 121, 122
TV_PICTURE_COLOR, 122
TV_PICTURE_CONTRAST, 122
TV_PICTURE_TINT, 122
TV_PICTURE_VARCOUNT, 122, 123
TV_SERVICE_CONTROL, 123
TV_SERVICE_PICTURE, 123
TV_SERVICE_SERVCOUNT, 123
tv_service_table, 151, 152
tvc_varname, 152
TvCtrlPointCommandLoop, 134
TvCtrlPointPrintCommands, 134
TvCtrlPointPrintLongHelp, 134
TvCtrlPointPrintShortHelp, 134
TvCtrlPointStart, 134
TvCtrlPointTimerLoopRun, 152
TvCtrlPointVerifyTimeouts, 135
TvDeviceCallbackEventHandler, 135
TvDeviceCommandLoop, 136
TvDeviceDecreaseBrightness, 136
TvDeviceDecreaseChannel, 136
TvDeviceDecreaseColor, 138
TvDeviceDecreaseContrast, 138
TvDeviceDecreaseTint, 138
TvDeviceDecreaseVolume, 139
TvDeviceHandleActionRequest, 139
TvDeviceHandleGetVarRequest, 140
TvDeviceHandleSubscriptionRequest, 140
TvDeviceIncreaseBrightness, 140
TvDeviceIncreaseChannel, 141
TvDeviceIncreaseColor, 141
TvDeviceIncreaseContrast, 141
TvDeviceIncreaseTint, 142
TvDeviceIncreaseVolume, 142
TvDevicePowerOff, 143
TvDevicePowerOn, 143
TvDeviceSetBrightness, 143
TvDeviceSetChannel, 144
TvDeviceSetColor, 144
TvDeviceSetContrast, 145
TvDeviceSetPower, 145
TvDeviceSetServiceTableVar, 145
TvDeviceSetTint, 146
TvDeviceSetVolume, 146
TvDeviceStart, 147

TvDeviceStateTableInit, 148
TvDeviceStop, 148
TvDeviceType, 152
TVDevMutex, 152
tvp_varname, 153
TvServiceName, 153
TvStateUpdate, 148
TvVarCount, 153
TvVarName, 153
upnp_action, 124
SampleUtil_FindAndParseService
Sample Code, 127
SampleUtil_Finish
Sample Code, 127
SampleUtil_GetElementValue
Sample Code, 128
SampleUtil_GetFirstDocumentItem
Sample Code, 128
SampleUtil_GetFirstElementItem
Sample Code, 128
SampleUtil_GetFirstServiceList
Sample Code, 129
SampleUtil_Initialize
Sample Code, 129
SampleUtil_Print
Sample Code, 130
SampleUtil_PrintEvent
Sample Code, 130
SampleUtil_PrintEventType
Sample Code, 131
SampleUtil_RegisterUpdateFunction
Sample Code, 131
SampleUtil_StateUpdate
Sample Code, 132
savePtr
_Parser, 231
scanner_t, 266
cursor, 267
entire_msg_loaded, 267
msg, 267
schedPolicy
THREADPOOLATTR, 282
ScheduleGenaAutoRenew
gena_ctrlpt.c, 666
search_extension
webserver.c, 713
SearchByTarget
SSDP Library, 170
seek
VirtualDirCallbacks, 297
send_action_response
soap_device.c, 849
send_error_response
soap_device.c, 850
send_search_result
SSDP Library, 170
send_var_query_response
soap_device.c, 850
SendInstruction, 267
Cookie, 268
CorsHeader, 268
ReadSendSize, 268
RecvWriteSize, 268
RequestCookie, 268
SendReply
SSDP Library, 171
SERVICE_INFO, 269
SERVICE_TABLE, 270
service_table.h
addServiceTable, 793
copy_subscription, 793
FindServiceControlURLPath, 794
FindServiceEventURLPath, 794
FindServiceId, 794
freeService, 795
freeServiceList, 795
freeServiceTable, 795
freeSubscription, 796
freeSubscriptionList, 796
getElementValue, 796
GetFirstSubscription, 796
GetNextSubscription, 797
getServiceTable, 797
getSubElement, 798
GetSubscriptionSID, 798
printService, 798
printServiceList, 799
printServiceTable, 799
removeServiceTable, 799
RemoveSubscriptionSID, 800
ServiceAdvertisement
SSDP Library, 171
ServiceId
TvService, 289
ServiceList
Handle_Info, 240
ServiceReply
SSDP Library, 171
ServiceShutdown
SSDP Library, 172
ServiceTable
Handle_Info, 240
ServiceType
TvService, 289
SetActionTable
Sample Code, 132
SetGenaCallback
miniserver.h, 786
SetHTTPGetCallback
miniserver.h, 786
SetPolicyType
ThreadPool.c, 885
SetPriority
ThreadPool.c, 885
SetRelTimeout
ThreadPool.c, 886

SetSeed
ThreadPool.c, 886
SetServiceTable
Sample Code, 132
SetSoapCallback
miniserver.h, 786
shutdown
THREADPOOL, 280
size
LINKEDLIST, 251
size_inc
membuffer, 254
soap_device.c
check_soap_request, 846
check_soapaction_hdr, 847
get_dev_service, 847
get_mpost_acton_hdrval, 848
handle_invoke_action, 848
handle_query_variable, 849
send_action_response, 849
send_error_response, 850
send_var_query_response, 850
soap_device_callback, 850
SOAP_TIMEOUT, 846
SOAP_URN, 851
soap_device_callback
soap_device.c, 850
soaplib.h, 804
soap_devserv_t, 271
SOAP_TIMEOUT
soap_device.c, 846
SOAP_URN
soap_device.c, 851
soaplib.h
soap_device_callback, 804
sock_close
Network Socket Library, 155
sock_destroy
Network Socket Library, 156
sock_init
Network Socket Library, 156
sock_init_with_ip
Network Socket Library, 157
sock_make_blocking
Network Socket Library, 157
sock_make_no_blocking
Network Socket Library, 158
sock_read
Network Socket Library, 158
sock_read_write
Network Socket Library, 158
sock_write
Network Socket Library, 159
SOCKET
Network Socket Library, 155
socket
Network Socket Library, 160
SOCKET_ERROR
Network Socket Library, 155
SOCKINFO, 271
SSDP Library, 160
AdvertiseAndReply, 165
advertiseAndReplyThread, 165
create_ssdp_sock_reqv4, 165
create_ssdp_sock_v4, 166
CreateClientRequestPacket, 166
CreateServicePacket, 166
DeviceAdvertisement, 167
DeviceReply, 167
DeviceShutdown, 168
extractIPv6address, 168
free_ssdp_event_handler_data, 168
get_ssdp_sockets, 169
isUrlV6UlaGua, 169
NewRequestHandler, 169
NO_ERROR_FOUND, 164
readFromSSDPSocket, 170
SearchByTarget, 170
send_search_result, 170
SendReply, 171
ServiceAdvertisement, 171
ServiceReply, 171
ServiceShutdown, 172
ssdp_event_handler_thread, 172
ssdp_handle_ctrlpt_msg, 173
ssdp_handle_device_request, 173
ssdp_request_type, 173
ssdp_request_type1, 174
SSDP_SERROR, 165
SsdpEvent, 164
ssdpReqSock, 175
SsdpSearchType, 164
ssdpSock, 175
start_event_handler, 174
SType, 164
unique_service_name, 174
valid_ssdp_msg, 175
ssdp_event_handler_thread
SSDP Library, 172
ssdp_handle_ctrlpt_msg
SSDP Library, 173
ssdp_handle_device_request
SSDP Library, 173
ssdp_request_type
SSDP Library, 173
ssdp_request_type1
SSDP Library, 174
SSDP_SERROR
SSDP Library, 165
ssdp_thread_data, 272
SsdpEvent
SSDP Library, 164
SsdpEventStruct, 272
ssdpReqSock
SSDP Library, 175
ssdpReqSock4

MServerSockArray, 257
ssdpReqSock6
MServerSockArray, 257
SSDPResultData
SSDPResultData.h, 860
SSDPResultData.c
SSDPResultData_assign, 857
SSDPResultData_delete, 857
SSDPResultData_dup, 857
SSDPResultData_get_Cookie, 857
SSDPResultData_get_CtrlptCallback, 857
SSDPResultData_get_Param, 857
SSDPResultData_new, 858
SSDPResultData_set_Cookie, 858
SSDPResultData_set_CtrlptCallback, 858
SSDPResultData_set_Param, 858
SSDPResultData.h
SSDPResultData, 860
SSDPResultData_assign, 860
SSDPResultData_delete, 860
SSDPResultData_dup, 861
SSDPResultData_get_Cookie, 861
SSDPResultData_get_CtrlptCallback, 861
SSDPResultData_get_Param, 861
SSDPResultData_new, 861
SSDPResultData_set_Cookie, 862
SSDPResultData_set_CtrlptCallback, 862
SSDPResultData_set_Param, 862
SSDPResultData_assign
SSDPResultData.c, 857
SSDPResultData.h, 860
SSDPResultData_delete
SSDPResultData.c, 857
SSDPResultData.h, 860
SSDPResultData_dup
SSDPResultData.c, 857
SSDPResultData.h, 861
SSDPResultData_get_Cookie
SSDPResultData.c, 857
SSDPResultData.h, 861
SSDPResultData_get_CtrlptCallback
SSDPResultData.c, 857
SSDPResultData.h, 861
SSDPResultData_get_Param
SSDPResultData.c, 857
SSDPResultData.h, 861
SSDPResultData_new
SSDPResultData.c, 858
SSDPResultData.h, 861
SSDPResultData_set_Cookie
SSDPResultData.c, 858
SSDPResultData.h, 862
SSDPResultData_set_CtrlptCallback
SSDPResultData.c, 858
SSDPResultData.h, 862
SSDPResultData_set_Param
SSDPResultData.c, 858
SSDPResultData.h, 862
ssdpsearcharg, 273
ssdpsearchexparg, 273
SsdpSearchList
Handle_Info, 240
ssdpsearchreply, 274
SsdpSearchType
SSDP Library, 164
ssdpSock
SSDP Library, 175
ssdpSock4
MServerSockArray, 257
ssdpSock6
MServerSockArray, 257
ssdpSock6UlaGua
MServerSockArray, 257
SSDPSockArray, 274
stackSize
THREADPOOLATTR, 282
start_and_shutdown
THREADPOOL, 280
start_event_handler
SSDP Library, 174
StartMiniServer
miniserver.c, 688
miniserver.h, 786
starvationTime
THREADPOOLATTR, 282
state_update
Sample Code, 124
STATS
ThreadPool.h, 902
stats
THREADPOOL, 280
StatsAccountHQ
ThreadPool.c, 886
StatsAccountLQ
ThreadPool.c, 886
StatsAccountMQ
ThreadPool.c, 887
StatsInit
ThreadPool.c, 887
StatsTime
ThreadPool.c, 887
status_code
http_message_t, 244
status_msg
http_message_t, 245
StopMiniServer
miniserver.c, 689
miniserver.h, 787
str_alloc
membuffer.h, 782
str_int_entry, 274
StrStr
webserver.c, 714
StrTok
webserver.c, 714
SType

SSDP Library, 164
SubscribeLock
gena.h, 739
SubscribeUnlock
gena.h, 739
SUBSCRIPTION, 275
SUpnpString, 275
sysdep.c
get_ieee_node_identifier, 928
get_system_time, 928
sysdep.h
get_ieee_node_identifier, 820
get_system_time, 820
UUIDS_PER_TICK, 820

tail
LINKEDLIST, 251
TData, 276
TestClass
TestClass.h, 396
TestClass.c
TestClass_add_to_list_TheList, 388
TestClass_assign, 388
TestClass_clear_TheBuffer, 389
TestClass_clear_TheString, 389
TestClass_delete, 389
TestClass_dup, 389
TestClass_get_TheActionComplete, 389
TestClass_get_TheBuffer, 390
TestClass_get_TheCharPointer, 390
TestClass_get_TheDomString, 390
TestClass_get_TheDomString_cstr, 390
TestClass_get_TheInteger, 390
TestClass_get_TheList, 390
TestClass_get_TheString, 391
TestClass_get_TheString_cstr, 391
TestClass_get_TheString_Length, 391
TestClass_get_TheUnsignedLong, 391
TestClass_new, 391
TestClass_set_TheActionComplete, 392
TestClass_set_TheBuffer, 392
TestClass_set_TheCharPointer, 392
TestClass_set_TheDomString, 392
TestClass_set_TheInteger, 392
TestClass_set_TheList, 393
TestClass_set_TheString, 393
TestClass_set_TheUnsignedLong, 393
TestClass_strcpy_TheString, 393
TestClass_strncpy_TheString, 393
TestClass.h
TestClass, 396
TestClass_add_to_list_TheList, 396
TestClass_assign, 396
TestClass_clear_TheBuffer, 396
TestClass_clear_TheString, 396
TestClass_delete, 397
TestClass_dup, 397
TestClass_get_TheActionComplete, 397
TestClass_get_TheBuffer, 397
TestClass_get_TheCharPointer, 397
TestClass_get_TheDomString, 397
TestClass_get_TheDomString_cstr, 398
TestClass_get_TheInteger, 398
TestClass_get_TheList, 398
TestClass_get_TheString, 398
TestClass_get_TheString_cstr, 398
TestClass_get_TheString_Length, 398
TestClass_get_TheUnsignedLong, 399
TestClass_new, 399
TestClass_set_TheActionComplete, 399
TestClass_set_TheBuffer, 399
TestClass_set_TheCharPointer, 399
TestClass_set_TheDomString, 400
TestClass_set_TheInteger, 400
TestClass_set_TheList, 400
TestClass_set_TheString, 400
TestClass_set_TheUnsignedLong, 400
TestClass_strcpy_TheString, 401
TestClass_strncpy_TheString, 401
TestClass_add_to_list_TheList
TestClass.c, 388
TestClass.h, 396
TestClass_assign
TestClass.c, 388
TestClass.h, 396
TestClass_clear_TheBuffer
TestClass.c, 389
TestClass.h, 396
TestClass_clear_TheString
TestClass.c, 389
TestClass.h, 396
TestClass_delete
TestClass.c, 389
TestClass.h, 397
TestClass_dup
TestClass.c, 389
TestClass.h, 397
TestClass_get_TheActionComplete
TestClass.c, 389
TestClass.h, 397
TestClass_get_TheBuffer
TestClass.c, 390
TestClass.h, 397
TestClass_get_TheCharPointer
TestClass.c, 390
TestClass.h, 397
TestClass_get_TheDomString
TestClass.c, 390
TestClass.h, 397
TestClass_get_TheDomString_cstr
TestClass.c, 390
TestClass.h, 398
TestClass_get_TheInteger
TestClass.c, 390
TestClass.h, 398
TestClass_get_TheList
TestClass.c, 390

TestClass.h, 398
TestClass_get_TheString
TestClass.c, 391
TestClass.h, 398
TestClass_get_TheString_cstr
TestClass.c, 391
TestClass.h, 398
TestClass_get_TheString_Length
TestClass.c, 391
TestClass.h, 398
TestClass_get_TheUnsignedLong
TestClass.c, 391
TestClass.h, 399
TestClass_new
TestClass.c, 391
TestClass.h, 399
TestClass_set_TheActionComplete
TestClass.c, 392
TestClass.h, 399
TestClass_set_TheBuffer
TestClass.c, 392
TestClass.h, 399
TestClass_set_TheCharPointer
TestClass.c, 392
TestClass.h, 399
TestClass_set_TheDomString
TestClass.c, 392
TestClass.h, 400
TestClass_set_TheInteger
TestClass.c, 392
TestClass.h, 400
TestClass_set_TheList
TestClass.c, 393
TestClass.h, 400
TestClass_set_TheString
TestClass.c, 393
TestClass.h, 400
TestClass_set_TheUnsignedLong
TestClass.c, 393
TestClass.h, 400
TestClass_strcpy_TheString
TestClass.c, 393
TestClass.h, 401
TestClass_strncpy_TheString
TestClass.c, 393
TestClass.h, 401
text
HOSTPORT, 241
The UpnpString Class, 97
m_length, 105
m_string, 105
UpnpString, 99
UpnpString_assign, 99
UpnpString_casecmp, 99
UpnpString_clear, 100
UpnpString_cmp, 100
UpnpString_delete, 101
UpnpString_dup, 101
UpnpString_get_Length, 101
UpnpString_get_String, 102
UpnpString_new, 103
UpnpString_set_Length, 103
UpnpString_set_String, 103
UpnpString_set_StringN, 104
TheStruct, 276
THREADPOOL, 277
attr, 278
busyThreads, 278
condition, 278
highJobQ, 278
jobFreeList, 278
lastJobId, 278
lowJobQ, 279
medJobQ, 279
mutex, 279
pendingWorkerThreadStart, 279
persistentJob, 279
persistentThreads, 279
shutdown, 280
start_and_shutdown, 280
stats, 280
totalThreads, 280
ThreadPool
ThreadPool.h, 902
ThreadPool.c
AddWorker, 882
BumpPriority, 882
CalcWaitTime, 882
CmpThreadPoolJob, 883
CreateThreadPoolJob, 883
CreateWorker, 884
DiffMillis, 884
FreeThreadPoolJob, 884
SetPolicyType, 885
SetPriority, 885
SetRelTimeout, 886
SetSeed, 886
StatsAccountHQ, 886
StatsAccountLQ, 886
StatsAccountMQ, 887
StatsInit, 887
StatsTime, 887
ThreadPoolAdd, 888
ThreadPoolAddPersistent, 888
ThreadPoolGetAttr, 889
ThreadPoolGetStats, 889
ThreadPoolInit, 890
ThreadPoolPrintStats, 890
ThreadPoolRemove, 891
ThreadPoolSetAttr, 891
ThreadPoolShutdown, 892
TPAttrInit, 892
TPAttrSetIdleTime, 892
TPAttrSetJobsPerThread, 893
TPAttrSetMaxJobsTotal, 893
TPAttrSetMaxThreads, 894

TPAttrSetMinThreads, 894
TPAttrSetSchedPolicy, 895
TPAttrSetStackSize, 895
TPAttrSetStarvationTime, 895
TPJobInit, 896
TPJobSetFreeFunction, 896
TPJobSetPriority, 897
TPSetMaxJobsTotal, 897
WorkerThread, 897
ThreadPool.h
DEFAULT_FREE_ROUTINE, 901
DEFAULT_IDLE_TIME, 901
DEFAULT_JOBS_PER_THREAD, 901
DEFAULT_MAX_JOBS_TOTAL, 901
DEFAULT_MAX_THREADS, 901
DEFAULT_MIN_THREADS, 901
DEFAULT_PRIORITY, 901
DEFAULT_STACK_SIZE, 901
DEFAULT_STARVATION_TIME, 902
free_routine, 902
INVALID_JOB_ID, 902
INVALID_POLICY, 902
JOBFREELISTSIZE, 902
STATS, 902
ThreadPool, 902
ThreadPoolAdd, 903
ThreadPoolAddPersistent, 904
ThreadPoolAttr, 903
ThreadPoolGetAttr, 904
ThreadPoolGetStats, 905
ThreadPoolInit, 905
ThreadPoolJob, 903
ThreadPoolPrintStats, 906
ThreadPoolRemove, 906
ThreadPoolSetAttr, 907
ThreadPoolShutdown, 907
ThreadPoolStats, 903
TPAttrInit, 908
TPAttrSetIdleTime, 908
TPAttrSetJobsPerThread, 908
TPAttrSetMaxJobsTotal, 909
TPAttrSetMaxThreads, 909
TPAttrSetMinThreads, 910
TPAttrSetSchedPolicy, 910
TPAttrSetStackSize, 911
TPAttrSetStarvationTime, 911
TPJobInit, 911
TPJobSetFreeFunction, 912
TPJobSetPriority, 912
TPSetMaxJobsTotal, 913
ThreadPoolAdd
ThreadPool.c, 888
ThreadPool.h, 903
ThreadPoolAddPersistent
ThreadPool.c, 888
ThreadPool.h, 904
THREADPOOLATTR, 281
jobsPerThread, 281
maxIdleTime, 281
maxJobsTotal, 281
maxThreads, 281
minThreads, 282
schedPolicy, 282
stackSize, 282
starvationTime, 282
ThreadPoolAttr
ThreadPool.h, 903
ThreadPoolGetAttr
ThreadPool.c, 889
ThreadPool.h, 904
ThreadPoolGetStats
ThreadPool.c, 889
ThreadPool.h, 905
ThreadPoolInit
ThreadPool.c, 890
ThreadPool.h, 905
THREADPOOLJOB, 282
ThreadPoolJob
ThreadPool.h, 903
ThreadPoolPrintStats
ThreadPool.c, 890
ThreadPool.h, 906
ThreadPoolRemove
ThreadPool.c, 891
ThreadPool.h, 906
ThreadPoolSetAttr
ThreadPool.c, 891
ThreadPool.h, 907
ThreadPoolShutdown
ThreadPool.c, 892
ThreadPool.h, 907
ThreadPoolStats
ThreadPool.h, 903
time_hi_and_version
_uuid_upnp, 232
time_low
_uuid_upnp, 233
time_mid
_uuid_upnp, 233
TimeoutType
TimerThread.h, 923
timeoutType
TimerThread.h, 923
TIMEREVENT, 283
eventTime, 284
persistent, 284
TimerEvent
TimerThread.h, 923
TIMERTHREAD, 284
TimerThread
TimerThread.h, 923
TimerThread.c
CalculateEventTime, 918
CreateTimerEvent, 918
FreeTimerEvent, 918
TimerThreadInit, 919

TimerThreadRemove, 919
TimerThreadSchedule, 920
TimerThreadShutdown, 920
TimerThreadWorker, 921
TimerThread.h
ABS_SEC, 924
REL_SEC, 924
TimeoutType, 923
timeoutType, 923
TimerEvent, 923
TimerThread, 923
TimerThreadInit, 924
TimerThreadRemove, 924
TimerThreadSchedule, 925
TimerThreadShutdown, 925
TimerThreadInit
TimerThread.c, 919
TimerThread.h, 924
TimerThreadRemove
TimerThread.c, 919
TimerThread.h, 924
TimerThreadSchedule
TimerThread.c, 920
TimerThread.h, 925
TimerThreadShutdown
TimerThread.c, 920
TimerThread.h, 925
TimerThreadWorker
TimerThread.c, 921
Todo List, 5
TOKEN, 285
token
uri.h, 833
token_cmp
uri.c, 728
uri.h, 838
token_string_casecmp
uri.c, 729
uri.h, 838
totalThreads
THREADPOOL, 280
ToUpperCase
webserver.c, 714
TPAttrInit
ThreadPool.c, 892
ThreadPool.h, 908
TPAttrSetIdleTime
ThreadPool.c, 892
ThreadPool.h, 908
TPAttrSetJobsPerThread
ThreadPool.c, 893
ThreadPool.h, 908
TPAttrSetMaxJobsTotal
ThreadPool.c, 893
ThreadPool.h, 909
TPAttrSetMaxThreads
ThreadPool.c, 894
ThreadPool.h, 909
TPAttrSetMinThreads
ThreadPool.c, 894
ThreadPool.h, 910
TPAttrSetSchedPolicy
ThreadPool.c, 895
ThreadPool.h, 910
TPAttrSetStackSize
ThreadPool.c, 895
ThreadPool.h, 911
TPAttrSetStarvationTime
ThreadPool.c, 895
ThreadPool.h, 911
TPJobInit
ThreadPool.c, 896
ThreadPool.h, 911
TPJobSetFreeFunction
ThreadPool.c, 896
ThreadPool.h, 912
TPJobSetPriority
ThreadPool.c, 897
ThreadPool.h, 912
TPOOLSTATS, 285
TPSetMaxJobsTotal
ThreadPool.c, 897
ThreadPool.h, 913
true_random
uuid.c, 931
ts
uuid_state, 295
TV_CONTROL_CHANNEL
Sample Code, 120
TV_CONTROL_POWER
Sample Code, 120
TV_CONTROL_VARCOUNT
Sample Code, 120
TV_CONTROL_VOLUME
Sample Code, 121
TV_MAX_VAL_LEN
Sample Code, 121
TV_MAXACTIONS
Sample Code, 121
TV_MAXVARS
Sample Code, 121
TV_PICTURE_BRIGHTNESS
Sample Code, 121, 122
TV_PICTURE_COLOR
Sample Code, 122
TV_PICTURE_CONTRAST
Sample Code, 122
TV_PICTURE_TINT
Sample Code, 122
TV_PICTURE_VARCOUNT
Sample Code, 122, 123
tv_service, 286
TV_SERVICE_CONTROL
Sample Code, 123
TV_SERVICE_PICTURE
Sample Code, 123

TV_SERVICE_SERVCOUNT
Sample Code, 123
tv_service_table
Sample Code, 151, 152
tvc_varname
Sample Code, 152
TvCtrlPointCommandLoop
Sample Code, 134
TvCtrlPointPrintCommands
Sample Code, 134
TvCtrlPointPrintLongHelp
Sample Code, 134
TvCtrlPointPrintShortHelp
Sample Code, 134
TvCtrlPointStart
Sample Code, 134
TvCtrlPointTimerLoopRun
Sample Code, 152
TvCtrlPointVerifyTimeouts
Sample Code, 135
TvDevice, 287
TvDeviceCallbackEventHandler
Sample Code, 135
TvDeviceCommandLoop
Sample Code, 136
TvDeviceDecreaseBrightness
Sample Code, 136
TvDeviceDecreaseChannel
Sample Code, 136
TvDeviceDecreaseColor
Sample Code, 138
TvDeviceDecreaseContrast
Sample Code, 138
TvDeviceDecreaseTint
Sample Code, 138
TvDeviceDecreaseVolume
Sample Code, 139
TvDeviceHandleActionRequest
Sample Code, 139
TvDeviceHandleGetVarRequest
Sample Code, 140
TvDeviceHandleSubscriptionRequest
Sample Code, 140
TvDeviceIncreaseBrightness
Sample Code, 140
TvDeviceIncreaseChannel
Sample Code, 141
TvDeviceIncreaseColor
Sample Code, 141
TvDeviceIncreaseContrast
Sample Code, 141
TvDeviceIncreaseTint
Sample Code, 142
TvDeviceIncreaseVolume
Sample Code, 142
TvDeviceNode, 287
TvDevicePowerOff
Sample Code, 143
TvDevicePowerOn
Sample Code, 143
TvDeviceSetBrightness
Sample Code, 143
TvDeviceSetChannel
Sample Code, 144
TvDeviceSetColor
Sample Code, 144
TvDeviceSetContrast
Sample Code, 145
TvDeviceSetPower
Sample Code, 145
TvDeviceSetServiceTableVar
Sample Code, 145
TvDeviceSetTint
Sample Code, 146
TvDeviceSetVolume
Sample Code, 146
TvDeviceStart
Sample Code, 147
TvDeviceStateTableInit
Sample Code, 148
TvDeviceStop
Sample Code, 148
TvDeviceType
Sample Code, 152
TVDevMutex
Sample Code, 152
tvp_varname
Sample Code, 153
TvService, 288
ActionNames, 289
actions, 289
ServiceId, 289
ServiceType, 289
UDN, 289
VariableCount, 289
VariableName, 289
VariableStrVal, 289
TvServiceName
Sample Code, 153
TvStateUpdate
Sample Code, 148
TvVarCount
Sample Code, 153
TvVarName
Sample Code, 153

UDN
TvService, 289
unique_service_name
SSDP Library, 174
UPnP API, 17
AutoAdvertise, 36
bWebServerState, 91
free_action_arg, 36
free_advertise_arg, 36
FreeHandle, 37
g_maxContentLength, 91

g_UpnpSdkEQMaxAge, 91
g_UpnpSdkEQMaxLen, 92
gAllowLiteralHostRedirection, 92
gDocumentRootDir, 92
GetCallBackFn, 37
GetClientHandleInfo, 37
GetDescDocumentAndURL, 38
GetDeviceHandleInfo, 38
GetDeviceHandleInfoForPath, 39
GetFreeHandle, 39
GetHandleInfo, 39
gIF_INDEX, 92
gIF_IPV4, 92
gIF_IPV4_NETMASK, 92
gIF_IPV6, 93
gIF_IPV6_PREFIX_LENGTH, 93
gIF_IPV6_ULA_GUA, 93
gIF_IPV6_ULA_GUA_PREFIX_LENGTH, 93
gIF_NAME, 93
GlobalClientSubscribeMutex, 93
GlobalHndRWLock, 94
gMiniServerThreadPool, 94
gRecvThreadPool, 94
gSDKInitMutex, 94
gSendThreadPool, 94
gTimerThread, 94
gUUIDMutex, 95
gWebCallback_HostValidate, 95
gWebCallback_HostValidateCookie, 95
gWebserverCorsString, 95
HandleTable, 95
IN6_IS_ADDR_GLOBAL, 26
IN6_IS_ADDR_ULA, 26
LOCAL_PORT_V4, 95
LOCAL_PORT_V6, 96
LOCAL_PORT_V6_ULA_GUA, 96
PrintHandleInfo, 40
PrintThreadPoolStats, 40
pVirtualDirList, 96
Upnp_DescType_e, 35
UPNP_E_ALREADY_REGISTERED, 26
UPNP_E_BAD_HTTPMSG, 27
UPNP_E_BAD_RESPONSE, 27
UPNP_E_CANCELED, 27
UPNP_E_EXT_NOT_XML, 27
UPNP_E_FILE_NOT_FOUND, 27
UPNP_E_FILE_READ_ERROR, 27
UPNP_E_FINISH, 28
UPNP_E_INIT, 28
UPNP_E_INIT_FAILED, 28
UPNP_E_INTERNAL_ERROR, 28
UPNP_E_INVALID_ACTION, 28
UPNP_E_INVALID_ARGUMENT, 28
UPNP_E_INVALID_DESC, 29
UPNP_E_INVALID_HANDLE, 29
UPNP_E_INVALID_INTERFACE, 29
UPNP_E_INVALID_PARAM, 29
UPNP_E_INVALID_SERVICE, 29
UPNP_E_INVALID_URL, 29
UPNP_E_LISTEN, 30
UPNP_E_NETWORK_ERROR, 30
UPNP_E_NOT_FOUND, 30
UPNP_E_NOTIFY_UNACCEPTED, 30
UPNP_E_OUTOF_HANDLE, 30
UPNP_E_OUTOF_MEMORY, 30
UPNP_E_OUTOF_SOCKET, 31
UPNP_E_SOCKET_BIND, 31
UPNP_E_SOCKET_CONNECT, 31
UPNP_E_SOCKET_ERROR, 31
UPNP_E_SOCKET_READ, 31
UPNP_E_SOCKET_WRITE, 32
UPNP_E_SUBSCRIBE_UNACCEPTED, 32
UPNP_E_SUCCESS, 32
UPNP_E_TIMEDOUT, 32
UPNP_E_UNSUBSCRIBE_UNACCEPTED, 32
UPNP_E_URL_TOO_BIG, 32
Upnp_HttpMethod_e, 36
UPNP_S_ALL, 36
UPNP_S_DEVICE, 36
UPNP_S_ROOT, 36
UPNP_S_SERVICE, 36
Upnp_SID, 33
Upnp_SType_e, 36
UpnpAcceptSubscription, 41
UpnpAcceptSubscriptionExt, 41
UpnpAddVirtualDir, 42
UpnpCancelHttpGet, 43
UpnpClient_Handle, 33
UpnpCloseHttpConnection, 43
UpnpCloseHttpGet, 44
UpnpCloseHttpPost, 44
UpnpComputeIpv6PrefixLength, 45
UpnpDevice_Handle, 33
UpnpDownloadUrlItem, 45
UpnpDownloadXmlDoc, 46
UpnpEnableWebserver, 46
UpnpEndHttpRequest, 47
UpnpFinish, 48
UpnpGetHttpResponse, 48
UpnpGetIfInfo, 49
UpnpGetServerIp6Address, 50
UpnpGetServerIpAddress, 51
UpnpGetServerPort, 51
UpnpGetServerPort6, 51
UpnpGetServerUlaGuaIp6Address, 52
UpnpGetServerUlaGuaPort6, 52
UpnpGetServiceVarStatus, 52
UpnpGetServiceVarStatusAsync, 53
UpnpHttpGetProgress, 54
UpnpInit2, 54
UpnpInitMutexes, 55
UpnpInitPreamble, 56
UpnpInitStartServers, 56
UpnpInitThreadPools, 57
UpnpIsWebserverEnabled, 57
UpnpMakeHttpRequest, 57

UpnpNotify, 58
UpnpNotifyExt, 59
UpnpOpenHttpConnection, 60
UpnpOpenHttpGet, 61
UpnpOpenHttpGetEx, 62
UpnpOpenHttpGetProxy, 63
UpnpOpenHttpPost, 63
UpnpReadHttpGet, 64
UpnpReadHttpResponse, 65
UPNPREG_BUF_DESC, 35
UPNPREG_FILENAME_DESC, 35
UPNPREG_URL_DESC, 35
UpnpRegisterClient, 66
UpnpRegisterRootDevice, 66
UpnpRegisterRootDevice2, 67
UpnpRegisterRootDevice3, 69
UpnpRegisterRootDevice4, 70
UpnpRemoveAllVirtualDirs, 71
UpnpRemoveVirtualDir, 71
UpnpRenewSubscription, 71
UpnpRenewSubscriptionAsync, 72
UpnpSdkClientRegistered, 96
UpnpSdkDeviceRegisteredV4, 96
UpnpSdkDeviceregisteredV6, 96
UpnpSdkInit, 97
UpnpSearchAsync, 73
UpnpSendAction, 74
UpnpSendActionAsync, 75
UpnpSendActionEx, 76
UpnpSendActionExAsync, 76
UpnpSendAdvertisement, 77
UpnpSendAdvertisementLowPower, 78
UpnpSetContentLength, 79
UpnpSetMaxContentLength, 79
UpnpSetMaxSubscriptions, 80
UpnpSetMaxSubscriptionTimeOut, 80
UpnpSetWebServerCorsString, 81
UpnpSetWebServerRootDir, 81
UpnpSubscribe, 82
UpnpSubscribeAsync, 83
UpnpThreadDistribution, 84
UpnpUnRegisterClient, 84
UpnpUnRegisterRootDevice, 84
UpnpUnRegisterRootDeviceLowPower, 85
UpnpUnSubscribe, 86
UpnpUnSubscribeAsync, 87
UpnpVirtualDir_set_CloseCallback, 88
UpnpVirtualDir_set_GetInfoCallback, 88
UpnpVirtualDir_set_OpenCallback, 88
UpnpVirtualDir_set_ReadCallback, 88
UpnpVirtualDir_set_SeekCallback, 89
UpnpVirtualDir_set_WriteCallback, 89
UpnpWebFileHandle, 33
UpnpWriteHttpPost, 89
UpnpWriteHttpRequest, 90
VDCallback_Close, 33
VDCallback_GetInfo, 34
VDCallback_Open, 34
VDCallback_Read, 34
VDCallback_Seek, 35
VDCallback_Write, 35
virtualDirCallback, 97
WinsockInit, 91
upnp_action
Sample Code, 124
UPNP_CONTROL_ACTION_COMPLETE
Callback.h, 404
UPNP_CONTROL_ACTION_REQUEST
Callback.h, 404
UPNP_CONTROL_GET_VAR_COMPLETE
Callback.h, 404
UPNP_CONTROL_GET_VAR_REQUEST
Callback.h, 404
UPNP_DEFAULT_LOG_LEVEL
upnpdebug.h, 456
Upnp_DescType_e
UPnP API, 35
UPNP_DISCOVERY_ADVERTISEMENT_ALIVE
Callback.h, 404
UPNP_DISCOVERY_ADVERTISEMENT_BYEBYE
Callback.h, 404
UPNP_DISCOVERY_SEARCH_RESULT
Callback.h, 404
UPNP_DISCOVERY_SEARCH_TIMEOUT
Callback.h, 404
UPNP_E_ALREADY_REGISTERED
UPnP API, 26
UPNP_E_BAD_HTTPMSG
UPnP API, 27
UPNP_E_BAD_RESPONSE
UPnP API, 27
UPNP_E_CANCELED
UPnP API, 27
UPNP_E_EXT_NOT_XML
UPnP API, 27
UPNP_E_FILE_NOT_FOUND
UPnP API, 27
UPNP_E_FILE_READ_ERROR
UPnP API, 27
UPNP_E_FINISH
UPnP API, 28
UPNP_E_INIT
UPnP API, 28
UPNP_E_INIT_FAILED
UPnP API, 28
UPNP_E_INTERNAL_ERROR
UPnP API, 28
UPNP_E_INVALID_ACTION
UPnP API, 28
UPNP_E_INVALID_ARGUMENT
UPnP API, 28
UPNP_E_INVALID_DESC
UPnP API, 29
UPNP_E_INVALID_HANDLE
UPnP API, 29
UPNP_E_INVALID_INTERFACE

UPnP API, 29
UPNP_E_INVALID_PARAM
UPnP API, 29
UPNP_E_INVALID_SERVICE
UPnP API, 29
UPNP_E_INVALID_URL
UPnP API, 29
UPNP_E_LISTEN
UPnP API, 30
UPNP_E_NETWORK_ERROR
UPnP API, 30
UPNP_E_NOT_FOUND
UPnP API, 30
UPNP_E_NOTIFY_UNACCEPTED
UPnP API, 30
UPNP_E_OUTOF_HANDLE
UPnP API, 30
UPNP_E_OUTOF_MEMORY
UPnP API, 30
UPNP_E_OUTOF_SOCKET
UPnP API, 31
UPNP_E_SOCKET_BIND
UPnP API, 31
UPNP_E_SOCKET_CONNECT
UPnP API, 31
UPNP_E_SOCKET_ERROR
UPnP API, 31
UPNP_E_SOCKET_READ
UPnP API, 31
UPNP_E_SOCKET_WRITE
UPnP API, 32
UPNP_E_SUBSCRIBE_UNACCEPTED
UPnP API, 32
UPNP_E_SUCCESS
UPnP API, 32
UPNP_E_TIMEDOUT
UPnP API, 32
UPNP_E_UNSUBSCRIBE_UNACCEPTED
UPnP API, 32
UPNP_E_URL_TOO_BIG
UPnP API, 32
UPNP_EVENT_AUTORENEWAL_FAILED
Callback.h, 405
UPNP_EVENT_RECEIVED
Callback.h, 404
UPNP_EVENT_RENEWAL_COMPLETE
Callback.h, 405
UPNP_EVENT_SUBSCRIBE_COMPLETE
Callback.h, 405
UPNP_EVENT_SUBSCRIPTION_EXPIRED
Callback.h, 405
UPNP_EVENT_SUBSCRIPTION_REQUEST
Callback.h, 404
UPNP_EVENT_UNSUBSCRIBE_COMPLETE
Callback.h, 405
Upnp_EventType_e
Callback.h, 403
UPNP_EXPORT_SPEC
UpnpGlobal.h, 508
Upnp_FunPtr
Callback.h, 403
Upnp_HttpMethod_e
UPnP API, 36
UPNP_INLINE
UpnpGlobal.h, 508
UPNP_S_ALL
UPnP API, 36
UPNP_S_DEVICE
UPnP API, 36
UPNP_S_ROOT
UPnP API, 36
UPNP_S_SERVICE
UPnP API, 36
Upnp_SID
UPnP API, 33
Upnp_SType_e
UPnP API, 36
upnp_uuid_unpack
uuid.c, 931
UpnpAcceptSubscription
UPnP API, 41
UpnpAcceptSubscriptionExt
UPnP API, 41
UpnpActionComplete
UpnpActionComplete.h, 436
UpnpActionComplete.c
UpnpActionComplete_assign, 568
UpnpActionComplete_clear_CtrlUrl, 568
UpnpActionComplete_delete, 568
UpnpActionComplete_dup, 568
UpnpActionComplete_get_ActionRequest, 569
UpnpActionComplete_get_ActionResult, 569
UpnpActionComplete_get_CtrlUrl, 569
UpnpActionComplete_get_CtrlUrl_cstr, 569
UpnpActionComplete_get_CtrlUrl_Length, 569
UpnpActionComplete_get_ErrCode, 570
UpnpActionComplete_new, 570
UpnpActionComplete_set_ActionRequest, 570
UpnpActionComplete_set_ActionResult, 570
UpnpActionComplete_set_CtrlUrl, 570
UpnpActionComplete_set_ErrCode, 571
UpnpActionComplete_strcpy_CtrlUrl, 571
UpnpActionComplete_strncpy_CtrlUrl, 571
UpnpActionComplete.h
UpnpActionComplete, 436
UpnpActionComplete_assign, 436
UpnpActionComplete_clear_CtrlUrl, 436
UpnpActionComplete_delete, 437
UpnpActionComplete_dup, 437
UpnpActionComplete_get_ActionRequest, 437
UpnpActionComplete_get_ActionResult, 437
UpnpActionComplete_get_CtrlUrl, 437
UpnpActionComplete_get_CtrlUrl_cstr, 438
UpnpActionComplete_get_CtrlUrl_Length, 438
UpnpActionComplete_get_ErrCode, 438
UpnpActionComplete_new, 438

UpnpActionComplete_set_ActionRequest, 438
UpnpActionComplete_set_ActionResult, 439
UpnpActionComplete_set_CtrlUrl, 439
UpnpActionComplete_set_ErrCode, 439
UpnpActionComplete_strcpy_CtrlUrl, 439
UpnpActionComplete_strncpy_CtrlUrl, 439
UpnpActionComplete_assign
UpnpActionComplete.c, 568
UpnpActionComplete.h, 436
UpnpActionComplete_clear_CtrlUrl
UpnpActionComplete.c, 568
UpnpActionComplete.h, 436
UpnpActionComplete_delete
UpnpActionComplete.c, 568
UpnpActionComplete.h, 437
UpnpActionComplete_dup
UpnpActionComplete.c, 568
UpnpActionComplete.h, 437
UpnpActionComplete_get_ActionRequest
UpnpActionComplete.c, 569
UpnpActionComplete.h, 437
UpnpActionComplete_get_ActionResult
UpnpActionComplete.c, 569
UpnpActionComplete.h, 437
UpnpActionComplete_get_CtrlUrl
UpnpActionComplete.c, 569
UpnpActionComplete.h, 437
UpnpActionComplete_get_CtrlUrl_cstr
UpnpActionComplete.c, 569
UpnpActionComplete.h, 438
UpnpActionComplete_get_CtrlUrl_Length
UpnpActionComplete.c, 569
UpnpActionComplete.h, 438
UpnpActionComplete_get_ErrCode
UpnpActionComplete.c, 570
UpnpActionComplete.h, 438
UpnpActionComplete_new
UpnpActionComplete.c, 570
UpnpActionComplete.h, 438
UpnpActionComplete_set_ActionRequest
UpnpActionComplete.c, 570
UpnpActionComplete.h, 438
UpnpActionComplete_set_ActionResult
UpnpActionComplete.c, 570
UpnpActionComplete.h, 439
UpnpActionComplete_set_CtrlUrl
UpnpActionComplete.c, 570
UpnpActionComplete.h, 439
UpnpActionComplete_set_ErrCode
UpnpActionComplete.c, 571
UpnpActionComplete.h, 439
UpnpActionComplete_strcpy_CtrlUrl
UpnpActionComplete.c, 571
UpnpActionComplete.h, 439
UpnpActionComplete_strncpy_CtrlUrl
UpnpActionComplete.c, 571
UpnpActionComplete.h, 439
UpnpActionRequest
UpnpActionRequest.h, 443
UpnpActionRequest.c
UpnpActionRequest_assign, 573
UpnpActionRequest_clear_ActionName, 573
UpnpActionRequest_clear_CtrlPtIPAddr, 574
UpnpActionRequest_clear_DevUDN, 574
UpnpActionRequest_clear_ErrStr, 574
UpnpActionRequest_clear_Os, 574
UpnpActionRequest_clear_ServiceID, 574
UpnpActionRequest_delete, 574
UpnpActionRequest_dup, 575
UpnpActionRequest_get_ActionName, 575
UpnpActionRequest_get_ActionName_cstr, 575
UpnpActionRequest_get_ActionName_Length, 575
UpnpActionRequest_get_ActionRequest, 575
UpnpActionRequest_get_ActionResult, 576
UpnpActionRequest_get_CtrlPtIPAddr, 576
UpnpActionRequest_get_DevUDN, 576
UpnpActionRequest_get_DevUDN_cstr, 576
UpnpActionRequest_get_DevUDN_Length, 576
UpnpActionRequest_get_ErrCode, 577
UpnpActionRequest_get_ErrStr, 577
UpnpActionRequest_get_ErrStr_cstr, 577
UpnpActionRequest_get_ErrStr_Length, 577
UpnpActionRequest_get_Os, 577
UpnpActionRequest_get_Os_cstr, 578
UpnpActionRequest_get_Os_Length, 578
UpnpActionRequest_get_ServiceID, 578
UpnpActionRequest_get_ServiceID_cstr, 578
UpnpActionRequest_get_ServiceID_Length, 578
UpnpActionRequest_get_SoapHeader, 579
UpnpActionRequest_get_Socket, 579
UpnpActionRequest_new, 579
UpnpActionRequest_set_ActionName, 579
UpnpActionRequest_set_ActionRequest, 579
UpnpActionRequest_set_ActionResult, 580
UpnpActionRequest_set_CtrlPtIPAddr, 580
UpnpActionRequest_set_DevUDN, 580
UpnpActionRequest_set_ErrCode, 580
UpnpActionRequest_set_ErrStr, 580
UpnpActionRequest_set_Os, 581
UpnpActionRequest_set_ServiceID, 581
UpnpActionRequest_set_SoapHeader, 581
UpnpActionRequest_set_Socket, 581
UpnpActionRequest_strcpy_ActionName, 581
UpnpActionRequest_strcpy_DevUDN, 582
UpnpActionRequest_strcpy_ErrStr, 582
UpnpActionRequest_strcpy_Os, 582
UpnpActionRequest_strcpy_ServiceID, 582
UpnpActionRequest_strncpy_ActionName, 583
UpnpActionRequest_strncpy_DevUDN, 583
UpnpActionRequest_strncpy_ErrStr, 583
UpnpActionRequest_strncpy_Os, 583
UpnpActionRequest_strncpy_ServiceID, 583
UpnpActionRequest.h
UpnpActionRequest, 443
UpnpActionRequest_assign, 443

UpnpActionRequest_clear_ActionName, 443
UpnpActionRequest_clear_CtrlPtIPAddr, 443
UpnpActionRequest_clear_DevUDN, 443
UpnpActionRequest_clear_ErrStr, 443
UpnpActionRequest_clear_Os, 444
UpnpActionRequest_clear_ServiceID, 444
UpnpActionRequest_delete, 444
UpnpActionRequest_dup, 444
UpnpActionRequest_get_ActionName, 444
UpnpActionRequest_get_ActionName_cstr, 445
UpnpActionRequest_get_ActionName_Length, 445
UpnpActionRequest_get_ActionRequest, 445
UpnpActionRequest_get_ActionResult, 445
UpnpActionRequest_get_CtrlPtIPAddr, 445
UpnpActionRequest_get_DevUDN, 446
UpnpActionRequest_get_DevUDN_cstr, 446
UpnpActionRequest_get_DevUDN_Length, 446
UpnpActionRequest_get_ErrCode, 446
UpnpActionRequest_get_ErrStr, 446
UpnpActionRequest_get_ErrStr_cstr, 447
UpnpActionRequest_get_ErrStr_Length, 447
UpnpActionRequest_get_Os, 447
UpnpActionRequest_get_Os_cstr, 447
UpnpActionRequest_get_Os_Length, 447
UpnpActionRequest_get_ServiceID, 448
UpnpActionRequest_get_ServiceID_cstr, 448
UpnpActionRequest_get_ServiceID_Length, 448
UpnpActionRequest_get_SoapHeader, 448
UpnpActionRequest_get_Socket, 448
UpnpActionRequest_new, 449
UpnpActionRequest_set_ActionName, 449
UpnpActionRequest_set_ActionRequest, 449
UpnpActionRequest_set_ActionResult, 449
UpnpActionRequest_set_CtrlPtIPAddr, 449
UpnpActionRequest_set_DevUDN, 450
UpnpActionRequest_set_ErrCode, 450
UpnpActionRequest_set_ErrStr, 450
UpnpActionRequest_set_Os, 450
UpnpActionRequest_set_ServiceID, 450
UpnpActionRequest_set_SoapHeader, 451
UpnpActionRequest_set_Socket, 451
UpnpActionRequest_strcpy_ActionName, 451
UpnpActionRequest_strcpy_DevUDN, 451
UpnpActionRequest_strcpy_ErrStr, 451
UpnpActionRequest_strcpy_Os, 452
UpnpActionRequest_strcpy_ServiceID, 452
UpnpActionRequest_strncpy_ActionName, 452
UpnpActionRequest_strncpy_DevUDN, 452
UpnpActionRequest_strncpy_ErrStr, 452
UpnpActionRequest_strncpy_Os, 453
UpnpActionRequest_strncpy_ServiceID, 453
UpnpActionRequest_assign
UpnpActionRequest.c, 573
UpnpActionRequest.h, 443
UpnpActionRequest_clear_ActionName
UpnpActionRequest.c, 573
UpnpActionRequest.h, 443
UpnpActionRequest_clear_CtrlPtIPAddr
UpnpActionRequest.c, 574
UpnpActionRequest.h, 443
UpnpActionRequest_clear_DevUDN
UpnpActionRequest.c, 574
UpnpActionRequest.h, 443
UpnpActionRequest_clear_ErrStr
UpnpActionRequest.c, 574
UpnpActionRequest.h, 443
UpnpActionRequest_clear_Os
UpnpActionRequest.c, 574
UpnpActionRequest.h, 444
UpnpActionRequest_clear_ServiceID
UpnpActionRequest.c, 574
UpnpActionRequest.h, 444
UpnpActionRequest_delete
UpnpActionRequest.c, 574
UpnpActionRequest.h, 444
UpnpActionRequest_dup
UpnpActionRequest.c, 575
UpnpActionRequest.h, 444
UpnpActionRequest_get_ActionName
UpnpActionRequest.c, 575
UpnpActionRequest.h, 444
UpnpActionRequest_get_ActionName_cstr
UpnpActionRequest.c, 575
UpnpActionRequest.h, 445
UpnpActionRequest_get_ActionName_Length
UpnpActionRequest.c, 575
UpnpActionRequest.h, 445
UpnpActionRequest_get_ActionRequest
UpnpActionRequest.c, 575
UpnpActionRequest.h, 445
UpnpActionRequest_get_ActionResult
UpnpActionRequest.c, 576
UpnpActionRequest.h, 445
UpnpActionRequest_get_CtrlPtIPAddr
UpnpActionRequest.c, 576
UpnpActionRequest.h, 445
UpnpActionRequest_get_DevUDN
UpnpActionRequest.c, 576
UpnpActionRequest.h, 446
UpnpActionRequest_get_DevUDN_cstr
UpnpActionRequest.c, 576
UpnpActionRequest.h, 446
UpnpActionRequest_get_DevUDN_Length
UpnpActionRequest.c, 576
UpnpActionRequest.h, 446
UpnpActionRequest_get_ErrCode
UpnpActionRequest.c, 577
UpnpActionRequest.h, 446
UpnpActionRequest_get_ErrStr
UpnpActionRequest.c, 577
UpnpActionRequest.h, 446
UpnpActionRequest_get_ErrStr_cstr
UpnpActionRequest.c, 577
UpnpActionRequest.h, 447
UpnpActionRequest_get_ErrStr_Length

UpnpActionRequest.c, 577
UpnpActionRequest.h, 447
UpnpActionRequest_get_Os
UpnpActionRequest.c, 577
UpnpActionRequest.h, 447
UpnpActionRequest_get_Os_cstr
UpnpActionRequest.c, 578
UpnpActionRequest.h, 447
UpnpActionRequest_get_Os_Length
UpnpActionRequest.c, 578
UpnpActionRequest.h, 447
UpnpActionRequest_get_ServiceID
UpnpActionRequest.c, 578
UpnpActionRequest.h, 448
UpnpActionRequest_get_ServiceID_cstr
UpnpActionRequest.c, 578
UpnpActionRequest.h, 448
UpnpActionRequest_get_ServiceID_Length
UpnpActionRequest.c, 578
UpnpActionRequest.h, 448
UpnpActionRequest_get_SoapHeader
UpnpActionRequest.c, 579
UpnpActionRequest.h, 448
UpnpActionRequest_get_Socket
UpnpActionRequest.c, 579
UpnpActionRequest.h, 448
UpnpActionRequest_new
UpnpActionRequest.c, 579
UpnpActionRequest.h, 449
UpnpActionRequest_set_ActionName
UpnpActionRequest.c, 579
UpnpActionRequest.h, 449
UpnpActionRequest_set_ActionRequest
UpnpActionRequest.c, 579
UpnpActionRequest.h, 449
UpnpActionRequest_set_ActionResult
UpnpActionRequest.c, 580
UpnpActionRequest.h, 449
UpnpActionRequest_set_CtrlPtIPAddr
UpnpActionRequest.c, 580
UpnpActionRequest.h, 449
UpnpActionRequest_set_DevUDN
UpnpActionRequest.c, 580
UpnpActionRequest.h, 450
UpnpActionRequest_set_ErrCode
UpnpActionRequest.c, 580
UpnpActionRequest.h, 450
UpnpActionRequest_set_ErrStr
UpnpActionRequest.c, 580
UpnpActionRequest.h, 450
UpnpActionRequest_set_Os
UpnpActionRequest.c, 581
UpnpActionRequest.h, 450
UpnpActionRequest_set_ServiceID
UpnpActionRequest.c, 581
UpnpActionRequest.h, 450
UpnpActionRequest_set_SoapHeader
UpnpActionRequest.c, 581
UpnpActionRequest.h, 451
UpnpActionRequest_set_Socket
UpnpActionRequest.c, 581
UpnpActionRequest.h, 451
UpnpActionRequest_strcpy_ActionName
UpnpActionRequest.c, 581
UpnpActionRequest.h, 451
UpnpActionRequest_strcpy_DevUDN
UpnpActionRequest.c, 582
UpnpActionRequest.h, 451
UpnpActionRequest_strcpy_ErrStr
UpnpActionRequest.c, 582
UpnpActionRequest.h, 451
UpnpActionRequest_strcpy_Os
UpnpActionRequest.c, 582
UpnpActionRequest.h, 452
UpnpActionRequest_strcpy_ServiceID
UpnpActionRequest.c, 582
UpnpActionRequest.h, 452
UpnpActionRequest_strncpy_ActionName
UpnpActionRequest.c, 583
UpnpActionRequest.h, 452
UpnpActionRequest_strncpy_DevUDN
UpnpActionRequest.c, 583
UpnpActionRequest.h, 452
UpnpActionRequest_strncpy_ErrStr
UpnpActionRequest.c, 583
UpnpActionRequest.h, 452
UpnpActionRequest_strncpy_Os
UpnpActionRequest.c, 583
UpnpActionRequest.h, 453
UpnpActionRequest_strncpy_ServiceID
UpnpActionRequest.c, 583
UpnpActionRequest.h, 453
UpnpAddToAction
Optional Tool API, 106
UpnpAddToActionResponse
Optional Tool API, 107
UpnpAddToPropertySet
Optional Tool API, 107
UpnpAddVirtualDir
UPnP API, 42
upnpapi.h
gUpnpSdkNLSuuid, 825
HandleReadLock, 824
HandleUnlock, 824
HandleWriteLock, 824
UpnpCancelHttpGet
UPnP API, 43
UpnpClient_Handle
UPnP API, 33
UpnpCloseHttpConnection
UPnP API, 43
UpnpCloseHttpGet
UPnP API, 44
UpnpCloseHttpPost
UPnP API, 44
UpnpCloseLog

upnpdebug.c, 591
upnpdebug.h, 456
UpnpCloseSocket
Network Socket Library, 155
UpnpComputeIpv6PrefixLength
UPnP API, 45
UpnpCreatePropertySet
Optional Tool API, 108
upnpdebug.c
g_log_level, 593
GlobalDebugMutex, 593
UpnpCloseLog, 591
UpnpGetDebugFile, 591
UpnpInitLog, 591
UpnpPrintf, 592
UpnpSetLogFileNames, 593
UpnpSetLogLevel, 593
upnpdebug.h
UPNP_DEFAULT_LOG_LEVEL, 456
UpnpCloseLog, 456
UpnpGetDebugFile, 456
UpnpInitLog, 457
UpnpPrintf, 457
UpnpSetLogFileNames, 458
UpnpSetLogLevel, 458
UpnpDevice_Handle
UPnP API, 33
UpnpDiscovery
UpnpDiscovery.h, 463
UpnpDiscovery.c
UpnpDiscovery_assign, 596
UpnpDiscovery_clear_Date, 596
UpnpDiscovery_clear_DestAddr, 596
UpnpDiscovery_clear_DeviceID, 596
UpnpDiscovery_clear_DeviceType, 596
UpnpDiscovery_clear_Ext, 597
UpnpDiscovery_clear_Location, 597
UpnpDiscovery_clear_Os, 597
UpnpDiscovery_clear_ServiceType, 597
UpnpDiscovery_clear_ServiceVer, 597
UpnpDiscovery_delete, 597
UpnpDiscovery_dup, 598
UpnpDiscovery_get_Date, 598
UpnpDiscovery_get_Date_cstr, 598
UpnpDiscovery_get_Date_Length, 598
UpnpDiscovery_get_DestAddr, 598
UpnpDiscovery_get_DeviceID, 599
UpnpDiscovery_get_DeviceID_cstr, 599
UpnpDiscovery_get_DeviceID_Length, 599
UpnpDiscovery_get_DeviceType, 599
UpnpDiscovery_get_DeviceType_cstr, 599
UpnpDiscovery_get_DeviceType_Length, 600
UpnpDiscovery_get_ErrCode, 600
UpnpDiscovery_get_Expires, 600
UpnpDiscovery_get_Ext, 600
UpnpDiscovery_get_Ext_cstr, 600
UpnpDiscovery_get_Ext_Length, 600
UpnpDiscovery_get_Location, 601
UpnpDiscovery_get_Location_cstr, 601
UpnpDiscovery_get_Location_Length, 601
UpnpDiscovery_get_Os, 601
UpnpDiscovery_get_Os_cstr, 601
UpnpDiscovery_get_Os_Length, 602
UpnpDiscovery_get_ServiceType, 602
UpnpDiscovery_get_ServiceType_cstr, 602
UpnpDiscovery_get_ServiceType_Length, 602
UpnpDiscovery_get_ServiceVer, 602
UpnpDiscovery_get_ServiceVer_cstr, 603
UpnpDiscovery_get_ServiceVer_Length, 603
UpnpDiscovery_new, 603
UpnpDiscovery_set_Date, 603
UpnpDiscovery_set_DestAddr, 603
UpnpDiscovery_set_DeviceID, 604
UpnpDiscovery_set_DeviceType, 604
UpnpDiscovery_set_ErrCode, 604
UpnpDiscovery_set_Expires, 604
UpnpDiscovery_set_Ext, 604
UpnpDiscovery_set_Location, 605
UpnpDiscovery_set_Os, 605
UpnpDiscovery_set_ServiceType, 605
UpnpDiscovery_set_ServiceVer, 605
UpnpDiscovery_strcpy_Date, 606
UpnpDiscovery_strcpy_DeviceID, 606
UpnpDiscovery_strcpy_DeviceType, 606
UpnpDiscovery_strcpy_Ext, 606
UpnpDiscovery_strcpy_Location, 607
UpnpDiscovery_strcpy_Os, 607
UpnpDiscovery_strcpy_ServiceType, 607
UpnpDiscovery_strcpy_ServiceVer, 607
UpnpDiscovery_strncpy_Date, 607
UpnpDiscovery_strncpy_DeviceID, 608
UpnpDiscovery_strncpy_DeviceType, 608
UpnpDiscovery_strncpy_Ext, 608
UpnpDiscovery_strncpy_Location, 608
UpnpDiscovery_strncpy_Os, 609
UpnpDiscovery_strncpy_ServiceType, 609
UpnpDiscovery_strncpy_ServiceVer, 609
UpnpDiscovery.h
UpnpDiscovery, 463
UpnpDiscovery_assign, 463
UpnpDiscovery_clear_Date, 463
UpnpDiscovery_clear_DestAddr, 463
UpnpDiscovery_clear_DeviceID, 464
UpnpDiscovery_clear_DeviceType, 464
UpnpDiscovery_clear_Ext, 464
UpnpDiscovery_clear_Location, 464
UpnpDiscovery_clear_Os, 464
UpnpDiscovery_clear_ServiceType, 464
UpnpDiscovery_clear_ServiceVer, 465
UpnpDiscovery_delete, 465
UpnpDiscovery_dup, 465
UpnpDiscovery_get_Date, 465
UpnpDiscovery_get_Date_cstr, 465
UpnpDiscovery_get_Date_Length, 466
UpnpDiscovery_get_DestAddr, 466
UpnpDiscovery_get_DeviceID, 466

UpnpDiscovery_get_DeviceID_cstr, 466
UpnpDiscovery_get_DeviceID_Length, 466
UpnpDiscovery_get_DeviceType, 467
UpnpDiscovery_get_DeviceType_cstr, 467
UpnpDiscovery_get_DeviceType_Length, 467
UpnpDiscovery_get_ErrCode, 467
UpnpDiscovery_get_Expires, 467
UpnpDiscovery_get_Ext, 468
UpnpDiscovery_get_Ext_cstr, 468
UpnpDiscovery_get_Ext_Length, 468
UpnpDiscovery_get_Location, 468
UpnpDiscovery_get_Location_cstr, 468
UpnpDiscovery_get_Location_Length, 469
UpnpDiscovery_get_Os, 469
UpnpDiscovery_get_Os_cstr, 469
UpnpDiscovery_get_Os_Length, 469
UpnpDiscovery_get_ServiceType, 469
UpnpDiscovery_get_ServiceType_cstr, 470
UpnpDiscovery_get_ServiceType_Length, 470
UpnpDiscovery_get_ServiceVer, 470
UpnpDiscovery_get_ServiceVer_cstr, 470
UpnpDiscovery_get_ServiceVer_Length, 470
UpnpDiscovery_new, 471
UpnpDiscovery_set_Date, 471
UpnpDiscovery_set_DestAddr, 471
UpnpDiscovery_set_DeviceID, 471
UpnpDiscovery_set_DeviceType, 471
UpnpDiscovery_set_ErrCode, 472
UpnpDiscovery_set_Expires, 472
UpnpDiscovery_set_Ext, 472
UpnpDiscovery_set_Location, 472
UpnpDiscovery_set_Os, 472
UpnpDiscovery_set_ServiceType, 473
UpnpDiscovery_set_ServiceVer, 473
UpnpDiscovery_strcpy_Date, 473
UpnpDiscovery_strcpy_DeviceID, 473
UpnpDiscovery_strcpy_DeviceType, 474
UpnpDiscovery_strcpy_Ext, 474
UpnpDiscovery_strcpy_Location, 474
UpnpDiscovery_strcpy_Os, 474
UpnpDiscovery_strcpy_ServiceType, 474
UpnpDiscovery_strcpy_ServiceVer, 475
UpnpDiscovery_strncpy_Date, 475
UpnpDiscovery_strncpy_DeviceID, 475
UpnpDiscovery_strncpy_DeviceType, 475
UpnpDiscovery_strncpy_Ext, 475
UpnpDiscovery_strncpy_Location, 476
UpnpDiscovery_strncpy_Os, 476
UpnpDiscovery_strncpy_ServiceType, 476
UpnpDiscovery_strncpy_ServiceVer, 476
UpnpDiscovery_assign
UpnpDiscovery.c, 596
UpnpDiscovery.h, 463
UpnpDiscovery_clear_Date
UpnpDiscovery.c, 596
UpnpDiscovery.h, 463
UpnpDiscovery_clear_DestAddr
UpnpDiscovery.c, 596
UpnpDiscovery.h, 463
UpnpDiscovery_clear_DeviceID
UpnpDiscovery.c, 596
UpnpDiscovery.h, 464
UpnpDiscovery_clear_DeviceType
UpnpDiscovery.c, 596
UpnpDiscovery.h, 464
UpnpDiscovery_clear_Ext
UpnpDiscovery.c, 597
UpnpDiscovery.h, 464
UpnpDiscovery_clear_Location
UpnpDiscovery.c, 597
UpnpDiscovery.h, 464
UpnpDiscovery_clear_Os
UpnpDiscovery.c, 597
UpnpDiscovery.h, 464
UpnpDiscovery_clear_ServiceType
UpnpDiscovery.c, 597
UpnpDiscovery.h, 464
UpnpDiscovery_clear_ServiceVer
UpnpDiscovery.c, 597
UpnpDiscovery.h, 465
UpnpDiscovery_delete
UpnpDiscovery.c, 597
UpnpDiscovery.h, 465
UpnpDiscovery_dup
UpnpDiscovery.c, 598
UpnpDiscovery.h, 465
UpnpDiscovery_get_Date
UpnpDiscovery.c, 598
UpnpDiscovery.h, 465
UpnpDiscovery_get_Date_cstr
UpnpDiscovery.c, 598
UpnpDiscovery.h, 465
UpnpDiscovery_get_Date_Length
UpnpDiscovery.c, 598
UpnpDiscovery.h, 466
UpnpDiscovery_get_DestAddr
UpnpDiscovery.c, 598
UpnpDiscovery.h, 466
UpnpDiscovery_get_DeviceID
UpnpDiscovery.c, 599
UpnpDiscovery.h, 466
UpnpDiscovery_get_DeviceID_cstr
UpnpDiscovery.c, 599
UpnpDiscovery.h, 466
UpnpDiscovery_get_DeviceID_Length
UpnpDiscovery.c, 599
UpnpDiscovery.h, 466
UpnpDiscovery_get_DeviceType
UpnpDiscovery.c, 599
UpnpDiscovery.h, 467
UpnpDiscovery_get_DeviceType_cstr
UpnpDiscovery.c, 599
UpnpDiscovery.h, 467
UpnpDiscovery_get_DeviceType_Length
UpnpDiscovery.c, 600
UpnpDiscovery.h, 467

UpnpDiscovery_get_ErrCode
UpnpDiscovery.c, 600
UpnpDiscovery.h, 467
UpnpDiscovery_get_Expires
UpnpDiscovery.c, 600
UpnpDiscovery.h, 467
UpnpDiscovery_get_Ext
UpnpDiscovery.c, 600
UpnpDiscovery.h, 468
UpnpDiscovery_get_Ext_cstr
UpnpDiscovery.c, 600
UpnpDiscovery.h, 468
UpnpDiscovery_get_Ext_Length
UpnpDiscovery.c, 600
UpnpDiscovery.h, 468
UpnpDiscovery_get_Location
UpnpDiscovery.c, 601
UpnpDiscovery.h, 468
UpnpDiscovery_get_Location_cstr
UpnpDiscovery.c, 601
UpnpDiscovery.h, 468
UpnpDiscovery_get_Location_Length
UpnpDiscovery.c, 601
UpnpDiscovery.h, 469
UpnpDiscovery_get_Os
UpnpDiscovery.c, 601
UpnpDiscovery.h, 469
UpnpDiscovery_get_Os_cstr
UpnpDiscovery.c, 601
UpnpDiscovery.h, 469
UpnpDiscovery_get_Os_Length
UpnpDiscovery.c, 602
UpnpDiscovery.h, 469
UpnpDiscovery_get_ServiceType
UpnpDiscovery.c, 602
UpnpDiscovery.h, 469
UpnpDiscovery_get_ServiceType_cstr
UpnpDiscovery.c, 602
UpnpDiscovery.h, 470
UpnpDiscovery_get_ServiceType_Length
UpnpDiscovery.c, 602
UpnpDiscovery.h, 470
UpnpDiscovery_get_ServiceVer
UpnpDiscovery.c, 602
UpnpDiscovery.h, 470
UpnpDiscovery_get_ServiceVer_cstr
UpnpDiscovery.c, 603
UpnpDiscovery.h, 470
UpnpDiscovery_get_ServiceVer_Length
UpnpDiscovery.c, 603
UpnpDiscovery.h, 470
UpnpDiscovery_new
UpnpDiscovery.c, 603
UpnpDiscovery.h, 471
UpnpDiscovery_set_Date
UpnpDiscovery.c, 603
UpnpDiscovery.h, 471
UpnpDiscovery_set_DestAddr
UpnpDiscovery.c, 603
UpnpDiscovery.h, 471
UpnpDiscovery_set_DeviceID
UpnpDiscovery.c, 604
UpnpDiscovery.h, 471
UpnpDiscovery_set_DeviceType
UpnpDiscovery.c, 604
UpnpDiscovery.h, 471
UpnpDiscovery_set_ErrCode
UpnpDiscovery.c, 604
UpnpDiscovery.h, 472
UpnpDiscovery_set_Expires
UpnpDiscovery.c, 604
UpnpDiscovery.h, 472
UpnpDiscovery_set_Ext
UpnpDiscovery.c, 604
UpnpDiscovery.h, 472
UpnpDiscovery_set_Location
UpnpDiscovery.c, 605
UpnpDiscovery.h, 472
UpnpDiscovery_set_Os
UpnpDiscovery.c, 605
UpnpDiscovery.h, 472
UpnpDiscovery_set_ServiceType
UpnpDiscovery.c, 605
UpnpDiscovery.h, 473
UpnpDiscovery_set_ServiceVer
UpnpDiscovery.c, 605
UpnpDiscovery.h, 473
UpnpDiscovery_strcpy_Date
UpnpDiscovery.c, 606
UpnpDiscovery.h, 473
UpnpDiscovery_strcpy_DeviceID
UpnpDiscovery.c, 606
UpnpDiscovery.h, 473
UpnpDiscovery_strcpy_DeviceType
UpnpDiscovery.c, 606
UpnpDiscovery.h, 474
UpnpDiscovery_strcpy_Ext
UpnpDiscovery.c, 606
UpnpDiscovery.h, 474
UpnpDiscovery_strcpy_Location
UpnpDiscovery.c, 607
UpnpDiscovery.h, 474
UpnpDiscovery_strcpy_Os
UpnpDiscovery.c, 607
UpnpDiscovery.h, 474
UpnpDiscovery_strcpy_ServiceType
UpnpDiscovery.c, 607
UpnpDiscovery.h, 474
UpnpDiscovery_strcpy_ServiceVer
UpnpDiscovery.c, 607
UpnpDiscovery.h, 475
UpnpDiscovery_strncpy_Date
UpnpDiscovery.c, 607
UpnpDiscovery.h, 475
UpnpDiscovery_strncpy_DeviceID
UpnpDiscovery.c, 608

UpnpDiscovery.h, 475
UpnpDiscovery_strncpy_DeviceType
UpnpDiscovery.c, 608
UpnpDiscovery.h, 475
UpnpDiscovery_strncpy_Ext
UpnpDiscovery.c, 608
UpnpDiscovery.h, 475
UpnpDiscovery_strncpy_Location
UpnpDiscovery.c, 608
UpnpDiscovery.h, 476
UpnpDiscovery_strncpy_Os
UpnpDiscovery.c, 609
UpnpDiscovery.h, 476
UpnpDiscovery_strncpy_ServiceType
UpnpDiscovery.c, 609
UpnpDiscovery.h, 476
UpnpDiscovery_strncpy_ServiceVer
UpnpDiscovery.c, 609
UpnpDiscovery.h, 476
UpnpDownloadUrlItem
UPnP API, 45
UpnpDownloadXmlDoc
UPnP API, 46
UpnpEnableWebserver
UPnP API, 46
UpnpEndHttpRequest
UPnP API, 47
UpnpEvent
UpnpEvent.h, 480
UpnpEvent.c
UpnpEvent_assign, 611
UpnpEvent_clear_SID, 611
UpnpEvent_delete, 611
UpnpEvent_dup, 611
UpnpEvent_get_ChangedVariables, 611
UpnpEvent_get_EventKey, 612
UpnpEvent_get_SID, 612
UpnpEvent_get_SID_cstr, 612
UpnpEvent_get_SID_Length, 612
UpnpEvent_new, 612
UpnpEvent_set_ChangedVariables, 613
UpnpEvent_set_EventKey, 613
UpnpEvent_set_SID, 613
UpnpEvent_strcpy_SID, 613
UpnpEvent_strncpy_SID, 613
UpnpEvent.h
UpnpEvent, 480
UpnpEvent_assign, 480
UpnpEvent_clear_SID, 480
UpnpEvent_delete, 480
UpnpEvent_dup, 481
UpnpEvent_get_ChangedVariables, 481
UpnpEvent_get_EventKey, 481
UpnpEvent_get_SID, 481
UpnpEvent_get_SID_cstr, 481
UpnpEvent_get_SID_Length, 482
UpnpEvent_new, 482
UpnpEvent_set_ChangedVariables, 482
UpnpEvent_set_EventKey, 482
UpnpEvent_set_SID, 482
UpnpEvent_strcpy_SID, 483
UpnpEvent_strncpy_SID, 483
UpnpEvent_assign
UpnpEvent.c, 611
UpnpEvent.h, 480
UpnpEvent_clear_SID
UpnpEvent.c, 611
UpnpEvent.h, 480
UpnpEvent_delete
UpnpEvent.c, 611
UpnpEvent.h, 480
UpnpEvent_dup
UpnpEvent.c, 611
UpnpEvent.h, 481
UpnpEvent_get_ChangedVariables
UpnpEvent.c, 611
UpnpEvent.h, 481
UpnpEvent_get_EventKey
UpnpEvent.c, 612
UpnpEvent.h, 481
UpnpEvent_get_SID
UpnpEvent.c, 612
UpnpEvent.h, 481
UpnpEvent_get_SID_cstr
UpnpEvent.c, 612
UpnpEvent.h, 481
UpnpEvent_get_SID_Length
UpnpEvent.c, 612
UpnpEvent.h, 482
UpnpEvent_new
UpnpEvent.c, 612
UpnpEvent.h, 482
UpnpEvent_set_ChangedVariables
UpnpEvent.c, 613
UpnpEvent.h, 482
UpnpEvent_set_EventKey
UpnpEvent.c, 613
UpnpEvent.h, 482
UpnpEvent_set_SID
UpnpEvent.c, 613
UpnpEvent.h, 482
UpnpEvent_strcpy_SID
UpnpEvent.c, 613
UpnpEvent.h, 483
UpnpEvent_strncpy_SID
UpnpEvent.c, 613
UpnpEvent.h, 483
UpnpEventSubscribe
UpnpEventSubscribe.h, 485
UpnpEventSubscribe.c
UpnpEventSubscribe_assign, 615
UpnpEventSubscribe_clear_PublisherUrl, 615
UpnpEventSubscribe_clear_SID, 616
UpnpEventSubscribe_delete, 616
UpnpEventSubscribe_dup, 616
UpnpEventSubscribe_get_ErrCode, 616

UpnpEventSubscribe_get_PublisherUrl, 616
UpnpEventSubscribe_get_PublisherUrl_cstr, 617
UpnpEventSubscribe_get_PublisherUrl_Length, 617
UpnpEventSubscribe_get_SID, 617
UpnpEventSubscribe_get_SID_cstr, 617
UpnpEventSubscribe_get_SID_Length, 617
UpnpEventSubscribe_get_TimeOut, 618
UpnpEventSubscribe_new, 618
UpnpEventSubscribe_set_ErrCode, 618
UpnpEventSubscribe_set_PublisherUrl, 618
UpnpEventSubscribe_set_SID, 618
UpnpEventSubscribe_set_TimeOut, 619
UpnpEventSubscribe_strcpy_PublisherUrl, 619
UpnpEventSubscribe_strcpy_SID, 619
UpnpEventSubscribe_strncpy_PublisherUrl, 619
UpnpEventSubscribe_strncpy_SID, 620
UpnpEventSubscribe.h
UpnpEventSubscribe, 485
UpnpEventSubscribe_assign, 486
UpnpEventSubscribe_clear_PublisherUrl, 486
UpnpEventSubscribe_clear_SID, 486
UpnpEventSubscribe_delete, 486
UpnpEventSubscribe_dup, 486
UpnpEventSubscribe_get_ErrCode, 486
UpnpEventSubscribe_get_PublisherUrl, 487
UpnpEventSubscribe_get_PublisherUrl_cstr, 487
UpnpEventSubscribe_get_PublisherUrl_Length, 487
UpnpEventSubscribe_get_SID, 487
UpnpEventSubscribe_get_SID_cstr, 487
UpnpEventSubscribe_get_SID_Length, 488
UpnpEventSubscribe_get_TimeOut, 488
UpnpEventSubscribe_new, 488
UpnpEventSubscribe_set_ErrCode, 488
UpnpEventSubscribe_set_PublisherUrl, 488
UpnpEventSubscribe_set_SID, 489
UpnpEventSubscribe_set_TimeOut, 489
UpnpEventSubscribe_strcpy_PublisherUrl, 489
UpnpEventSubscribe_strcpy_SID, 489
UpnpEventSubscribe_strncpy_PublisherUrl, 490
UpnpEventSubscribe_strncpy_SID, 490
UpnpEventSubscribe_assign
UpnpEventSubscribe.c, 615
UpnpEventSubscribe.h, 486
UpnpEventSubscribe_clear_PublisherUrl
UpnpEventSubscribe.c, 615
UpnpEventSubscribe.h, 486
UpnpEventSubscribe_clear_SID
UpnpEventSubscribe.c, 616
UpnpEventSubscribe.h, 486
UpnpEventSubscribe_delete
UpnpEventSubscribe.c, 616
UpnpEventSubscribe.h, 486
UpnpEventSubscribe_dup
UpnpEventSubscribe.c, 616
UpnpEventSubscribe.h, 486
UpnpEventSubscribe_get_ErrCode
UpnpEventSubscribe.c, 616
UpnpEventSubscribe.h, 486
UpnpEventSubscribe_get_PublisherUrl
UpnpEventSubscribe.c, 616
UpnpEventSubscribe.h, 487
UpnpEventSubscribe_get_PublisherUrl_cstr
UpnpEventSubscribe.c, 617
UpnpEventSubscribe.h, 487
UpnpEventSubscribe_get_PublisherUrl_Length
UpnpEventSubscribe.c, 617
UpnpEventSubscribe.h, 487
UpnpEventSubscribe_get_SID
UpnpEventSubscribe.c, 617
UpnpEventSubscribe.h, 487
UpnpEventSubscribe_get_SID_cstr
UpnpEventSubscribe.c, 617
UpnpEventSubscribe.h, 487
UpnpEventSubscribe_get_SID_Length
UpnpEventSubscribe.c, 617
UpnpEventSubscribe.h, 488
UpnpEventSubscribe_get_TimeOut
UpnpEventSubscribe.c, 618
UpnpEventSubscribe.h, 488
UpnpEventSubscribe_new
UpnpEventSubscribe.c, 618
UpnpEventSubscribe.h, 488
UpnpEventSubscribe_set_ErrCode
UpnpEventSubscribe.c, 618
UpnpEventSubscribe.h, 488
UpnpEventSubscribe_set_PublisherUrl
UpnpEventSubscribe.c, 618
UpnpEventSubscribe.h, 488
UpnpEventSubscribe_set_SID
UpnpEventSubscribe.c, 618
UpnpEventSubscribe.h, 489
UpnpEventSubscribe_set_TimeOut
UpnpEventSubscribe.c, 619
UpnpEventSubscribe.h, 489
UpnpEventSubscribe_strcpy_PublisherUrl
UpnpEventSubscribe.c, 619
UpnpEventSubscribe.h, 489
UpnpEventSubscribe_strcpy_SID
UpnpEventSubscribe.c, 619
UpnpEventSubscribe.h, 489
UpnpEventSubscribe_strncpy_PublisherUrl
UpnpEventSubscribe.c, 619
UpnpEventSubscribe.h, 490
UpnpEventSubscribe_strncpy_SID
UpnpEventSubscribe.c, 620
UpnpEventSubscribe.h, 490
UpnpExtraHeaders
UpnpExtraHeaders.h, 493
UpnpExtraHeaders.c
UpnpExtraHeaders_add_to_list_node, 621
UpnpExtraHeaders_assign, 621
UpnpExtraHeaders_clear_name, 622
UpnpExtraHeaders_clear_value, 622
UpnpExtraHeaders_delete, 622

UpnpExtraHeaders_dup, 622
UpnpExtraHeaders_get_name, 622
UpnpExtraHeaders_get_name_cstr, 623
UpnpExtraHeaders_get_name_Length, 623
UpnpExtraHeaders_get_node, 623
UpnpExtraHeaders_get_resp, 623
UpnpExtraHeaders_get_resp_cstr, 623
UpnpExtraHeaders_get_value, 623
UpnpExtraHeaders_get_value_cstr, 624
UpnpExtraHeaders_get_value_Length, 624
UpnpExtraHeaders_new, 624
UpnpExtraHeaders_set_name, 624
UpnpExtraHeaders_set_node, 624
UpnpExtraHeaders_set_resp, 625
UpnpExtraHeaders_set_value, 625
UpnpExtraHeaders_strcpy_name, 625
UpnpExtraHeaders_strcpy_value, 625
UpnpExtraHeaders_strncpy_name, 625
UpnpExtraHeaders_strncpy_value, 626
UpnpExtraHeaders.h
UpnpExtraHeaders, 493
UpnpExtraHeaders_add_to_list_node, 493
UpnpExtraHeaders_assign, 493
UpnpExtraHeaders_clear_name, 494
UpnpExtraHeaders_clear_value, 494
UpnpExtraHeaders_delete, 494
UpnpExtraHeaders_dup, 494
UpnpExtraHeaders_get_name, 494
UpnpExtraHeaders_get_name_cstr, 495
UpnpExtraHeaders_get_name_Length, 495
UpnpExtraHeaders_get_node, 495
UpnpExtraHeaders_get_resp, 495
UpnpExtraHeaders_get_resp_cstr, 495
UpnpExtraHeaders_get_value, 495
UpnpExtraHeaders_get_value_cstr, 496
UpnpExtraHeaders_get_value_Length, 496
UpnpExtraHeaders_new, 496
UpnpExtraHeaders_set_name, 496
UpnpExtraHeaders_set_node, 496
UpnpExtraHeaders_set_resp, 497
UpnpExtraHeaders_set_value, 497
UpnpExtraHeaders_strcpy_name, 497
UpnpExtraHeaders_strcpy_value, 497
UpnpExtraHeaders_strncpy_name, 497
UpnpExtraHeaders_strncpy_value, 498
UpnpExtraHeaders_add_to_list_node
UpnpExtraHeaders.c, 621
UpnpExtraHeaders.h, 493
UpnpExtraHeaders_assign
UpnpExtraHeaders.c, 621
UpnpExtraHeaders.h, 493
UpnpExtraHeaders_clear_name
UpnpExtraHeaders.c, 622
UpnpExtraHeaders.h, 494
UpnpExtraHeaders_clear_value
UpnpExtraHeaders.c, 622
UpnpExtraHeaders.h, 494
UpnpExtraHeaders_delete
UpnpExtraHeaders.c, 622
UpnpExtraHeaders.h, 494
UpnpExtraHeaders_dup
UpnpExtraHeaders.c, 622
UpnpExtraHeaders.h, 494
UpnpExtraHeaders_get_name
UpnpExtraHeaders.c, 622
UpnpExtraHeaders.h, 494
UpnpExtraHeaders_get_name_cstr
UpnpExtraHeaders.c, 623
UpnpExtraHeaders.h, 495
UpnpExtraHeaders_get_name_Length
UpnpExtraHeaders.c, 623
UpnpExtraHeaders.h, 495
UpnpExtraHeaders_get_node
UpnpExtraHeaders.c, 623
UpnpExtraHeaders.h, 495
UpnpExtraHeaders_get_resp
UpnpExtraHeaders.c, 623
UpnpExtraHeaders.h, 495
UpnpExtraHeaders_get_resp_cstr
UpnpExtraHeaders.c, 623
UpnpExtraHeaders.h, 495
UpnpExtraHeaders_get_value
UpnpExtraHeaders.c, 623
UpnpExtraHeaders.h, 495
UpnpExtraHeaders_get_value_cstr
UpnpExtraHeaders.c, 624
UpnpExtraHeaders.h, 496
UpnpExtraHeaders_get_value_Length
UpnpExtraHeaders.c, 624
UpnpExtraHeaders.h, 496
UpnpExtraHeaders_new
UpnpExtraHeaders.c, 624
UpnpExtraHeaders.h, 496
UpnpExtraHeaders_set_name
UpnpExtraHeaders.c, 624
UpnpExtraHeaders.h, 496
UpnpExtraHeaders_set_node
UpnpExtraHeaders.c, 624
UpnpExtraHeaders.h, 496
UpnpExtraHeaders_set_resp
UpnpExtraHeaders.c, 625
UpnpExtraHeaders.h, 497
UpnpExtraHeaders_set_value
UpnpExtraHeaders.c, 625
UpnpExtraHeaders.h, 497
UpnpExtraHeaders_strcpy_name
UpnpExtraHeaders.c, 625
UpnpExtraHeaders.h, 497
UpnpExtraHeaders_strcpy_value
UpnpExtraHeaders.c, 625
UpnpExtraHeaders.h, 497
UpnpExtraHeaders_strncpy_name
UpnpExtraHeaders.c, 625
UpnpExtraHeaders.h, 497
UpnpExtraHeaders_strncpy_value
UpnpExtraHeaders.c, 626

UpnpExtraHeaders.h, 498
UpnpFileInfo
UpnpFileInfo.h, 501
UpnpFileInfo.c
UpnpFileInfo_add_to_list_ExtraHeadersList, 627
UpnpFileInfo_assign, 627
UpnpFileInfo_clear_CtrlPtIPAddr, 628
UpnpFileInfo_clear_Os, 628
UpnpFileInfo_delete, 628
UpnpFileInfo_dup, 628
UpnpFileInfo_get_ContentType, 628
UpnpFileInfo_get_ContentType_cstr, 629
UpnpFileInfo_get_CtrlPtIPAddr, 629
UpnpFileInfo_get_ExtraHeadersList, 629
UpnpFileInfo_get_FileLength, 629
UpnpFileInfo_get_IsDirectory, 629
UpnpFileInfo_get_IsReadable, 629
UpnpFileInfo_get_LastModified, 630
UpnpFileInfo_get_Os, 630
UpnpFileInfo_get_Os_cstr, 630
UpnpFileInfo_get_Os_Length, 630
UpnpFileInfo_new, 630
UpnpFileInfo_set_ContentType, 631
UpnpFileInfo_set_CtrlPtIPAddr, 631
UpnpFileInfo_set_ExtraHeadersList, 631
UpnpFileInfo_set_FileLength, 631
UpnpFileInfo_set_IsDirectory, 631
UpnpFileInfo_set_IsReadable, 632
UpnpFileInfo_set_LastModified, 632
UpnpFileInfo_set_Os, 632
UpnpFileInfo_strcpy_Os, 632
UpnpFileInfo_strncpy_Os, 632
UpnpFileInfo.h
UpnpFileInfo, 501
UpnpFileInfo_add_to_list_ExtraHeadersList, 501
UpnpFileInfo_assign, 501
UpnpFileInfo_clear_CtrlPtIPAddr, 501
UpnpFileInfo_clear_Os, 501
UpnpFileInfo_delete, 501
UpnpFileInfo_dup, 502
UpnpFileInfo_get_ContentType, 502
UpnpFileInfo_get_ContentType_cstr, 502
UpnpFileInfo_get_CtrlPtIPAddr, 502
UpnpFileInfo_get_ExtraHeadersList, 502
UpnpFileInfo_get_FileLength, 503
UpnpFileInfo_get_IsDirectory, 503
UpnpFileInfo_get_IsReadable, 503
UpnpFileInfo_get_LastModified, 503
UpnpFileInfo_get_Os, 503
UpnpFileInfo_get_Os_cstr, 503
UpnpFileInfo_get_Os_Length, 504
UpnpFileInfo_new, 504
UpnpFileInfo_set_ContentType, 504
UpnpFileInfo_set_CtrlPtIPAddr, 504
UpnpFileInfo_set_ExtraHeadersList, 504
UpnpFileInfo_set_FileLength, 505
UpnpFileInfo_set_IsDirectory, 505
UpnpFileInfo_set_IsReadable, 505
UpnpFileInfo_set_LastModified, 505
UpnpFileInfo_set_Os, 505
UpnpFileInfo_strcpy_Os, 506
UpnpFileInfo_strncpy_Os, 506
UpnpFileInfo_add_to_list_ExtraHeadersList
UpnpFileInfo.c, 627
UpnpFileInfo.h, 501
UpnpFileInfo_assign
UpnpFileInfo.c, 627
UpnpFileInfo.h, 501
UpnpFileInfo_clear_CtrlPtIPAddr
UpnpFileInfo.c, 628
UpnpFileInfo.h, 501
UpnpFileInfo_clear_Os
UpnpFileInfo.c, 628
UpnpFileInfo.h, 501
UpnpFileInfo_delete
UpnpFileInfo.c, 628
UpnpFileInfo.h, 501
UpnpFileInfo_dup
UpnpFileInfo.c, 628
UpnpFileInfo.h, 502
UpnpFileInfo_get_ContentType
UpnpFileInfo.c, 628
UpnpFileInfo.h, 502
UpnpFileInfo_get_ContentType_cstr
UpnpFileInfo.c, 629
UpnpFileInfo.h, 502
UpnpFileInfo_get_CtrlPtIPAddr
UpnpFileInfo.c, 629
UpnpFileInfo.h, 502
UpnpFileInfo_get_ExtraHeadersList
UpnpFileInfo.c, 629
UpnpFileInfo.h, 502
UpnpFileInfo_get_FileLength
UpnpFileInfo.c, 629
UpnpFileInfo.h, 503
UpnpFileInfo_get_IsDirectory
UpnpFileInfo.c, 629
UpnpFileInfo.h, 503
UpnpFileInfo_get_IsReadable
UpnpFileInfo.c, 629
UpnpFileInfo.h, 503
UpnpFileInfo_get_LastModified
UpnpFileInfo.c, 630
UpnpFileInfo.h, 503
UpnpFileInfo_get_Os
UpnpFileInfo.c, 630
UpnpFileInfo.h, 503
UpnpFileInfo_get_Os_cstr
UpnpFileInfo.c, 630
UpnpFileInfo.h, 503
UpnpFileInfo_get_Os_Length
UpnpFileInfo.c, 630
UpnpFileInfo.h, 504
UpnpFileInfo_new
UpnpFileInfo.c, 630
UpnpFileInfo.h, 504

UpnpFileInfo_set_ContentType
UpnpFileInfo.c, 631
UpnpFileInfo.h, 504
UpnpFileInfo_set_CtrlPtIPAddr
UpnpFileInfo.c, 631
UpnpFileInfo.h, 504
UpnpFileInfo_set_ExtraHeadersList
UpnpFileInfo.c, 631
UpnpFileInfo.h, 504
UpnpFileInfo_set_FileLength
UpnpFileInfo.c, 631
UpnpFileInfo.h, 505
UpnpFileInfo_set_IsDirectory
UpnpFileInfo.c, 631
UpnpFileInfo.h, 505
UpnpFileInfo_set_IsReadable
UpnpFileInfo.c, 632
UpnpFileInfo.h, 505
UpnpFileInfo_set_LastModified
UpnpFileInfo.c, 632
UpnpFileInfo.h, 505
UpnpFileInfo_set_Os
UpnpFileInfo.c, 632
UpnpFileInfo.h, 505
UpnpFileInfo_strcpy_Os
UpnpFileInfo.c, 632
UpnpFileInfo.h, 506
UpnpFileInfo_strncpy_Os
UpnpFileInfo.c, 632
UpnpFileInfo.h, 506
UpnpFinish
UPnP API, 48
UpnpGetDebugFile
upnpdebug.c, 591
upnpdebug.h, 456
UpnpGetErrorMessage
Optional Tool API, 108
UpnpGetHttpResponse
UPnP API, 48
UpnpGetIfInfo
UPnP API, 49
UpnpGetServerIp6Address
UPnP API, 50
UpnpGetServerIpAddress
UPnP API, 51
UpnpGetServerPort
UPnP API, 51
UpnpGetServerPort6
UPnP API, 51
UpnpGetServerUlaGuaIp6Address
UPnP API, 52
UpnpGetServerUlaGuaPort6
UPnP API, 52
UpnpGetServiceVarStatus
UPnP API, 52
UpnpGetServiceVarStatusAsync
UPnP API, 53
UpnpGlobal.h
PRIzd, 508
UPNP_EXPORT_SPEC, 508
UPNP_INLINE, 508
UpnpHttpGetProgress
UPnP API, 54
UpnpInit2
UPnP API, 54
UpnpInitLog
upnpdebug.c, 591
upnpdebug.h, 457
UpnpInitMutexes
UPnP API, 55
UpnpInitPreamble
UPnP API, 56
UpnpInitStartServers
UPnP API, 56
UpnpInitThreadPools
UPnP API, 57
UpnpIsWebserverEnabled
UPnP API, 57
UpnpListHead, 290
UpnpMakeAction
Optional Tool API, 109
UpnpMakeActionResponse
Optional Tool API, 109
UpnpMakeHttpRequest
UPnP API, 57
UpnpNonblockParam, 291
UpnpNotify
UPnP API, 58
UpnpNotifyExt
UPnP API, 59
UpnpOpenHttpConnection
UPnP API, 60
UpnpOpenHttpGet
UPnP API, 61
UpnpOpenHttpGetEx
UPnP API, 62
UpnpOpenHttpGetProxy
UPnP API, 63
UpnpOpenHttpPost
UPnP API, 63
UpnpPrintf
upnpdebug.c, 592
upnpdebug.h, 457
UpnpReadHttpGet
UPnP API, 64
UpnpReadHttpResponse
UPnP API, 65
UPNPREG_BUF_DESC
UPnP API, 35
UPNPREG_FILENAME_DESC
UPnP API, 35
UPNPREG_URL_DESC
UPnP API, 35
UpnpRegisterClient
UPnP API, 66
UpnpRegisterRootDevice

UPnP API, 66
UpnpRegisterRootDevice2
UPnP API, 67
UpnpRegisterRootDevice3
UPnP API, 69
UpnpRegisterRootDevice4
UPnP API, 70
UpnpRemoveAllVirtualDirs
UPnP API, 71
UpnpRemoveVirtualDir
UPnP API, 71
UpnpRenewSubscription
UPnP API, 71
UpnpRenewSubscriptionAsync
UPnP API, 72
UpnpResolveURL
Optional Tool API, 110
UpnpResolveURL2
Optional Tool API, 111
UpnpSdkClientRegistered
UPnP API, 96
UpnpSdkDeviceRegisteredV4
UPnP API, 96
UpnpSdkDeviceregisteredV6
UPnP API, 96
UpnpSdkInit
UPnP API, 97
UpnpSearchAsync
UPnP API, 73
UpnpSendAction
UPnP API, 74
UpnpSendActionAsync
UPnP API, 75
UpnpSendActionEx
UPnP API, 76
UpnpSendActionExAsync
UPnP API, 76
UpnpSendAdvertisement
UPnP API, 77
UpnpSendAdvertisementLowPower
UPnP API, 78
UpnpSetContentLength
UPnP API, 79
UpnpSetLogFileNames
upnpdebug.c, 593
upnpdebug.h, 458
UpnpSetLogLevel
upnpdebug.c, 593
upnpdebug.h, 458
UpnpSetMaxContentLength
UPnP API, 79
UpnpSetMaxJobsTotal
Optional Tool API, 111
UpnpSetMaxSubscriptions
UPnP API, 80
UpnpSetMaxSubscriptionTimeOut
UPnP API, 80
UpnpSetWebServerCorsString
UPnP API, 81
UpnpSetWebServerRootDir
UPnP API, 81
UpnpStateVarComplete
UpnpStateVarComplete.h, 513
UpnpStateVarComplete.c
UpnpStateVarComplete_assign, 634
UpnpStateVarComplete_clear_CtrlUrl, 634
UpnpStateVarComplete_clear_StateVarName, 635
UpnpStateVarComplete_delete, 635
UpnpStateVarComplete_dup, 635
UpnpStateVarComplete_get_CtrlUrl, 635
UpnpStateVarComplete_get_CtrlUrl_cstr, 635
UpnpStateVarComplete_get_CtrlUrl_Length, 636
UpnpStateVarComplete_get_CurrentVal, 636
UpnpStateVarComplete_get_CurrentVal_cstr, 636
UpnpStateVarComplete_get_ErrCode, 636
UpnpStateVarComplete_get_StateVarName, 636
UpnpStateVarComplete_get_StateVarName_cstr, 637
UpnpStateVarComplete_get_StateVarName_Length, 637
UpnpStateVarComplete_new, 637
UpnpStateVarComplete_set_CtrlUrl, 637
UpnpStateVarComplete_set_CurrentVal, 637
UpnpStateVarComplete_set_ErrCode, 638
UpnpStateVarComplete_set_StateVarName, 638
UpnpStateVarComplete_strcpy_CtrlUrl, 638
UpnpStateVarComplete_strcpy_StateVarName, 638
UpnpStateVarComplete_strncpy_CtrlUrl, 639
UpnpStateVarComplete_strncpy_StateVarName, 639
UpnpStateVarComplete.h
UpnpStateVarComplete, 513
UpnpStateVarComplete_assign, 514
UpnpStateVarComplete_clear_CtrlUrl, 514
UpnpStateVarComplete_clear_StateVarName, 514
UpnpStateVarComplete_delete, 514
UpnpStateVarComplete_dup, 514
UpnpStateVarComplete_get_CtrlUrl, 514
UpnpStateVarComplete_get_CtrlUrl_cstr, 515
UpnpStateVarComplete_get_CtrlUrl_Length, 515
UpnpStateVarComplete_get_CurrentVal, 515
UpnpStateVarComplete_get_CurrentVal_cstr, 515
UpnpStateVarComplete_get_ErrCode, 515
UpnpStateVarComplete_get_StateVarName, 516
UpnpStateVarComplete_get_StateVarName_cstr, 516
UpnpStateVarComplete_get_StateVarName_Length, 516
UpnpStateVarComplete_new, 516
UpnpStateVarComplete_set_CtrlUrl, 516
UpnpStateVarComplete_set_CurrentVal, 517
UpnpStateVarComplete_set_ErrCode, 517
UpnpStateVarComplete_set_StateVarName, 517
UpnpStateVarComplete_strcpy_CtrlUrl, 517

UpnpStateVarComplete_strcpy_StateVarName, 518
UpnpStateVarComplete_strncpy_CtrlUrl, 518
UpnpStateVarComplete_strncpy_StateVarName, 518
UpnpStateVarComplete_assign
UpnpStateVarComplete.c, 634
UpnpStateVarComplete.h, 514
UpnpStateVarComplete_clear_CtrlUrl
UpnpStateVarComplete.c, 634
UpnpStateVarComplete.h, 514
UpnpStateVarComplete_clear_StateVarName
UpnpStateVarComplete.c, 635
UpnpStateVarComplete.h, 514
UpnpStateVarComplete_delete
UpnpStateVarComplete.c, 635
UpnpStateVarComplete.h, 514
UpnpStateVarComplete_dup
UpnpStateVarComplete.c, 635
UpnpStateVarComplete.h, 514
UpnpStateVarComplete_get_CtrlUrl
UpnpStateVarComplete.c, 635
UpnpStateVarComplete.h, 514
UpnpStateVarComplete_get_CtrlUrl_cstr
UpnpStateVarComplete.c, 635
UpnpStateVarComplete.h, 515
UpnpStateVarComplete_get_CtrlUrl_Length
UpnpStateVarComplete.c, 636
UpnpStateVarComplete.h, 515
UpnpStateVarComplete_get_CurrentVal
UpnpStateVarComplete.c, 636
UpnpStateVarComplete.h, 515
UpnpStateVarComplete_get_CurrentVal_cstr
UpnpStateVarComplete.c, 636
UpnpStateVarComplete.h, 515
UpnpStateVarComplete_get_ErrCode
UpnpStateVarComplete.c, 636
UpnpStateVarComplete.h, 515
UpnpStateVarComplete_get_StateVarName
UpnpStateVarComplete.c, 636
UpnpStateVarComplete.h, 516
UpnpStateVarComplete_get_StateVarName_cstr
UpnpStateVarComplete.c, 637
UpnpStateVarComplete.h, 516
UpnpStateVarComplete_get_StateVarName_Length
UpnpStateVarComplete.c, 637
UpnpStateVarComplete.h, 516
UpnpStateVarComplete_new
UpnpStateVarComplete.c, 637
UpnpStateVarComplete.h, 516
UpnpStateVarComplete_set_CtrlUrl
UpnpStateVarComplete.c, 637
UpnpStateVarComplete.h, 516
UpnpStateVarComplete_set_CurrentVal
UpnpStateVarComplete.c, 637
UpnpStateVarComplete.h, 517
UpnpStateVarComplete_set_ErrCode
UpnpStateVarComplete.c, 638
UpnpStateVarComplete.h, 517
UpnpStateVarComplete_set_StateVarName
UpnpStateVarComplete.c, 638
UpnpStateVarComplete.h, 517
UpnpStateVarComplete_strcpy_CtrlUrl
UpnpStateVarComplete.c, 638
UpnpStateVarComplete.h, 517
UpnpStateVarComplete_strcpy_StateVarName
UpnpStateVarComplete.c, 638
UpnpStateVarComplete.h, 518
UpnpStateVarComplete_strncpy_CtrlUrl
UpnpStateVarComplete.c, 639
UpnpStateVarComplete.h, 518
UpnpStateVarComplete_strncpy_StateVarName
UpnpStateVarComplete.c, 639
UpnpStateVarComplete.h, 518
UpnpStateVarRequest
UpnpStateVarRequest.h, 521
UpnpStateVarRequest.c
UpnpStateVarRequest_assign, 641
UpnpStateVarRequest_clear_CtrlPtIPAddr, 641
UpnpStateVarRequest_clear_DevUDN, 641
UpnpStateVarRequest_clear_ErrStr, 641
UpnpStateVarRequest_clear_ServiceID, 641
UpnpStateVarRequest_clear_StateVarName, 642
UpnpStateVarRequest_delete, 642
UpnpStateVarRequest_dup, 642
UpnpStateVarRequest_get_CtrlPtIPAddr, 642
UpnpStateVarRequest_get_CurrentVal, 642
UpnpStateVarRequest_get_CurrentVal_cstr, 643
UpnpStateVarRequest_get_DevUDN, 643
UpnpStateVarRequest_get_DevUDN_cstr, 643
UpnpStateVarRequest_get_DevUDN_Length, 643
UpnpStateVarRequest_get_ErrCode, 643
UpnpStateVarRequest_get_ErrStr, 644
UpnpStateVarRequest_get_ErrStr_cstr, 644
UpnpStateVarRequest_get_ErrStr_Length, 644
UpnpStateVarRequest_get_ServiceID, 644
UpnpStateVarRequest_get_ServiceID_cstr, 644
UpnpStateVarRequest_get_ServiceID_Length, 645
UpnpStateVarRequest_get_Socket, 645
UpnpStateVarRequest_get_StateVarName, 645
UpnpStateVarRequest_get_StateVarName_cstr, 645
UpnpStateVarRequest_get_StateVarName_Length, 645
UpnpStateVarRequest_new, 646
UpnpStateVarRequest_set_CtrlPtIPAddr, 646
UpnpStateVarRequest_set_CurrentVal, 646
UpnpStateVarRequest_set_DevUDN, 646
UpnpStateVarRequest_set_ErrCode, 646
UpnpStateVarRequest_set_ErrStr, 647
UpnpStateVarRequest_set_ServiceID, 647
UpnpStateVarRequest_set_Socket, 647
UpnpStateVarRequest_set_StateVarName, 647
UpnpStateVarRequest_strcpy_DevUDN, 647
UpnpStateVarRequest_strcpy_ErrStr, 648

UpnpStateVarRequest_strcpy_ServiceID, 648
UpnpStateVarRequest_strcpy_StateVarName, 648
UpnpStateVarRequest_strncpy_DevUDN, 648
UpnpStateVarRequest_strncpy_ErrStr, 649
UpnpStateVarRequest_strncpy_ServiceID, 649
UpnpStateVarRequest_strncpy_StateVarName, 649
UpnpStateVarRequest.h
UpnpStateVarRequest, 521
UpnpStateVarRequest_assign, 522
UpnpStateVarRequest_clear_CtrlPtIPAddr, 522
UpnpStateVarRequest_clear_DevUDN, 522
UpnpStateVarRequest_clear_ErrStr, 522
UpnpStateVarRequest_clear_ServiceID, 522
UpnpStateVarRequest_clear_StateVarName, 522
UpnpStateVarRequest_delete, 523
UpnpStateVarRequest_dup, 523
UpnpStateVarRequest_get_CtrlPtIPAddr, 523
UpnpStateVarRequest_get_CurrentVal, 523
UpnpStateVarRequest_get_CurrentVal_cstr, 523
UpnpStateVarRequest_get_DevUDN, 524
UpnpStateVarRequest_get_DevUDN_cstr, 524
UpnpStateVarRequest_get_DevUDN_Length, 524
UpnpStateVarRequest_get_ErrCode, 524
UpnpStateVarRequest_get_ErrStr, 524
UpnpStateVarRequest_get_ErrStr_cstr, 525
UpnpStateVarRequest_get_ErrStr_Length, 525
UpnpStateVarRequest_get_ServiceID, 525
UpnpStateVarRequest_get_ServiceID_cstr, 525
UpnpStateVarRequest_get_ServiceID_Length, 525
UpnpStateVarRequest_get_Socket, 526
UpnpStateVarRequest_get_StateVarName, 526
UpnpStateVarRequest_get_StateVarName_cstr, 526
UpnpStateVarRequest_get_StateVarName_Length, 526
UpnpStateVarRequest_new, 526
UpnpStateVarRequest_set_CtrlPtIPAddr, 527
UpnpStateVarRequest_set_CurrentVal, 527
UpnpStateVarRequest_set_DevUDN, 527
UpnpStateVarRequest_set_ErrCode, 527
UpnpStateVarRequest_set_ErrStr, 527
UpnpStateVarRequest_set_ServiceID, 528
UpnpStateVarRequest_set_Socket, 528
UpnpStateVarRequest_set_StateVarName, 528
UpnpStateVarRequest_strcpy_DevUDN, 528
UpnpStateVarRequest_strcpy_ErrStr, 529
UpnpStateVarRequest_strcpy_ServiceID, 529
UpnpStateVarRequest_strcpy_StateVarName, 529
UpnpStateVarRequest_strncpy_DevUDN, 529
UpnpStateVarRequest_strncpy_ErrStr, 530
UpnpStateVarRequest_strncpy_ServiceID, 530
UpnpStateVarRequest_strncpy_StateVarName, 530
UpnpStateVarRequest_assign
UpnpStateVarRequest.c, 641
UpnpStateVarRequest.h, 522
UpnpStateVarRequest_clear_CtrlPtIPAddr
UpnpStateVarRequest.c, 641
UpnpStateVarRequest.h, 522
UpnpStateVarRequest_clear_DevUDN
UpnpStateVarRequest.c, 641
UpnpStateVarRequest.h, 522
UpnpStateVarRequest_clear_ErrStr
UpnpStateVarRequest.c, 641
UpnpStateVarRequest.h, 522
UpnpStateVarRequest_clear_ServiceID
UpnpStateVarRequest.c, 641
UpnpStateVarRequest.h, 522
UpnpStateVarRequest_clear_StateVarName
UpnpStateVarRequest.c, 642
UpnpStateVarRequest.h, 522
UpnpStateVarRequest_delete
UpnpStateVarRequest.c, 642
UpnpStateVarRequest.h, 523
UpnpStateVarRequest_dup
UpnpStateVarRequest.c, 642
UpnpStateVarRequest.h, 523
UpnpStateVarRequest_get_CtrlPtIPAddr
UpnpStateVarRequest.c, 642
UpnpStateVarRequest.h, 523
UpnpStateVarRequest_get_CurrentVal
UpnpStateVarRequest.c, 642
UpnpStateVarRequest.h, 523
UpnpStateVarRequest_get_CurrentVal_cstr
UpnpStateVarRequest.c, 643
UpnpStateVarRequest.h, 523
UpnpStateVarRequest_get_DevUDN
UpnpStateVarRequest.c, 643
UpnpStateVarRequest.h, 524
UpnpStateVarRequest_get_DevUDN_cstr
UpnpStateVarRequest.c, 643
UpnpStateVarRequest.h, 524
UpnpStateVarRequest_get_DevUDN_Length
UpnpStateVarRequest.c, 643
UpnpStateVarRequest.h, 524
UpnpStateVarRequest_get_ErrCode
UpnpStateVarRequest.c, 643
UpnpStateVarRequest.h, 524
UpnpStateVarRequest_get_ErrStr
UpnpStateVarRequest.c, 644
UpnpStateVarRequest.h, 524
UpnpStateVarRequest_get_ErrStr_cstr
UpnpStateVarRequest.c, 644
UpnpStateVarRequest.h, 525
UpnpStateVarRequest_get_ErrStr_Length
UpnpStateVarRequest.c, 644
UpnpStateVarRequest.h, 525
UpnpStateVarRequest_get_ServiceID
UpnpStateVarRequest.c, 644
UpnpStateVarRequest.h, 525
UpnpStateVarRequest_get_ServiceID_cstr
UpnpStateVarRequest.c, 644
UpnpStateVarRequest.h, 525
UpnpStateVarRequest_get_ServiceID_Length

UpnpStateVarRequest.c, 645
UpnpStateVarRequest.h, 525
UpnpStateVarRequest_get_Socket
UpnpStateVarRequest.c, 645
UpnpStateVarRequest.h, 526
UpnpStateVarRequest_get_StateVarName
UpnpStateVarRequest.c, 645
UpnpStateVarRequest.h, 526
UpnpStateVarRequest_get_StateVarName_cstr
UpnpStateVarRequest.c, 645
UpnpStateVarRequest.h, 526
UpnpStateVarRequest_get_StateVarName_Length
UpnpStateVarRequest.c, 645
UpnpStateVarRequest.h, 526
UpnpStateVarRequest_new
UpnpStateVarRequest.c, 646
UpnpStateVarRequest.h, 526
UpnpStateVarRequest_set_CtrlPtIPAddr
UpnpStateVarRequest.c, 646
UpnpStateVarRequest.h, 527
UpnpStateVarRequest_set_CurrentVal
UpnpStateVarRequest.c, 646
UpnpStateVarRequest.h, 527
UpnpStateVarRequest_set_DevUDN
UpnpStateVarRequest.c, 646
UpnpStateVarRequest.h, 527
UpnpStateVarRequest_set_ErrCode
UpnpStateVarRequest.c, 646
UpnpStateVarRequest.h, 527
UpnpStateVarRequest_set_ErrStr
UpnpStateVarRequest.c, 647
UpnpStateVarRequest.h, 527
UpnpStateVarRequest_set_ServiceID
UpnpStateVarRequest.c, 647
UpnpStateVarRequest.h, 528
UpnpStateVarRequest_set_Socket
UpnpStateVarRequest.c, 647
UpnpStateVarRequest.h, 528
UpnpStateVarRequest_set_StateVarName
UpnpStateVarRequest.c, 647
UpnpStateVarRequest.h, 528
UpnpStateVarRequest_strcpy_DevUDN
UpnpStateVarRequest.c, 647
UpnpStateVarRequest.h, 528
UpnpStateVarRequest_strcpy_ErrStr
UpnpStateVarRequest.c, 648
UpnpStateVarRequest.h, 529
UpnpStateVarRequest_strcpy_ServiceID
UpnpStateVarRequest.c, 648
UpnpStateVarRequest.h, 529
UpnpStateVarRequest_strcpy_StateVarName
UpnpStateVarRequest.c, 648
UpnpStateVarRequest.h, 529
UpnpStateVarRequest_strncpy_DevUDN
UpnpStateVarRequest.c, 648
UpnpStateVarRequest.h, 529
UpnpStateVarRequest_strncpy_ErrStr
UpnpStateVarRequest.c, 649
UpnpStateVarRequest.h, 530
UpnpStateVarRequest_strncpy_ServiceID
UpnpStateVarRequest.c, 649
UpnpStateVarRequest.h, 530
UpnpStateVarRequest_strncpy_StateVarName
UpnpStateVarRequest.c, 649
UpnpStateVarRequest.h, 530
UpnpString
The UpnpString Class, 99
UpnpString_assign
The UpnpString Class, 99
UpnpString_casecmp
The UpnpString Class, 99
UpnpString_clear
The UpnpString Class, 100
UpnpString_cmp
The UpnpString Class, 100
UpnpString_delete
The UpnpString Class, 101
UpnpString_dup
The UpnpString Class, 101
UpnpString_get_Length
The UpnpString Class, 101
UpnpString_get_String
The UpnpString Class, 102
UpnpString_new
The UpnpString Class, 103
UpnpString_set_Length
The UpnpString Class, 103
UpnpString_set_String
The UpnpString Class, 103
UpnpString_set_StringN
The UpnpString Class, 104
UpnpSubscribe
UPnP API, 82
UpnpSubscribeAsync
UPnP API, 83
UpnpSubscriptionRequest
UpnpSubscriptionRequest.h, 536
UpnpSubscriptionRequest.c
UpnpSubscriptionRequest_assign, 652
UpnpSubscriptionRequest_clear_ServiceId, 652
UpnpSubscriptionRequest_clear_SID, 653
UpnpSubscriptionRequest_clear_UDN, 653
UpnpSubscriptionRequest_delete, 653
UpnpSubscriptionRequest_dup, 653
UpnpSubscriptionRequest_get_ServiceId, 653
UpnpSubscriptionRequest_get_ServiceId_cstr, 654
UpnpSubscriptionRequest_get_ServiceId_Length, 654
UpnpSubscriptionRequest_get_SID, 654
UpnpSubscriptionRequest_get_SID_cstr, 654
UpnpSubscriptionRequest_get_SID_Length, 654
UpnpSubscriptionRequest_get_UDN, 655
UpnpSubscriptionRequest_get_UDN_cstr, 655
UpnpSubscriptionRequest_get_UDN_Length, 655
UpnpSubscriptionRequest_new, 655

UpnpSubscriptionRequest_set_ServiceId, 655
UpnpSubscriptionRequest_set_SID, 656
UpnpSubscriptionRequest_set_UDN, 656
UpnpSubscriptionRequest_strcpy_ServiceId, 656
UpnpSubscriptionRequest_strcpy_SID, 656
UpnpSubscriptionRequest_strcpy_UDN, 657
UpnpSubscriptionRequest_strncpy_ServiceId, 657
UpnpSubscriptionRequest_strncpy_SID, 657
UpnpSubscriptionRequest_strncpy_UDN, 657
UpnpSubscriptionRequest.h
UpnpSubscriptionRequest, 536
UpnpSubscriptionRequest_assign, 537
UpnpSubscriptionRequest_clear_ServiceId, 537
UpnpSubscriptionRequest_clear_SID, 537
UpnpSubscriptionRequest_clear_UDN, 537
UpnpSubscriptionRequest_delete, 537
UpnpSubscriptionRequest_dup, 537
UpnpSubscriptionRequest_get_ServiceId, 538
UpnpSubscriptionRequest_get_ServiceId_cstr, 538
UpnpSubscriptionRequest_get_ServiceId_Length, 538
UpnpSubscriptionRequest_get_SID, 538
UpnpSubscriptionRequest_get_SID_cstr, 538
UpnpSubscriptionRequest_get_SID_Length, 539
UpnpSubscriptionRequest_get_UDN, 539
UpnpSubscriptionRequest_get_UDN_cstr, 539
UpnpSubscriptionRequest_get_UDN_Length, 539
UpnpSubscriptionRequest_new, 539
UpnpSubscriptionRequest_set_ServiceId, 540
UpnpSubscriptionRequest_set_SID, 540
UpnpSubscriptionRequest_set_UDN, 540
UpnpSubscriptionRequest_strcpy_ServiceId, 540
UpnpSubscriptionRequest_strcpy_SID, 541
UpnpSubscriptionRequest_strcpy_UDN, 541
UpnpSubscriptionRequest_strncpy_ServiceId, 541
UpnpSubscriptionRequest_strncpy_SID, 541
UpnpSubscriptionRequest_strncpy_UDN, 542
UpnpSubscriptionRequest_assign
UpnpSubscriptionRequest.c, 652
UpnpSubscriptionRequest.h, 537
UpnpSubscriptionRequest_clear_ServiceId
UpnpSubscriptionRequest.c, 652
UpnpSubscriptionRequest.h, 537
UpnpSubscriptionRequest_clear_SID
UpnpSubscriptionRequest.c, 653
UpnpSubscriptionRequest.h, 537
UpnpSubscriptionRequest_clear_UDN
UpnpSubscriptionRequest.c, 653
UpnpSubscriptionRequest.h, 537
UpnpSubscriptionRequest_delete
UpnpSubscriptionRequest.c, 653
UpnpSubscriptionRequest.h, 537
UpnpSubscriptionRequest_dup
UpnpSubscriptionRequest.c, 653
UpnpSubscriptionRequest.h, 537
UpnpSubscriptionRequest_get_ServiceId
UpnpSubscriptionRequest.c, 653
UpnpSubscriptionRequest.h, 538
UpnpSubscriptionRequest_get_ServiceId_cstr
UpnpSubscriptionRequest.c, 654
UpnpSubscriptionRequest.h, 538
UpnpSubscriptionRequest_get_ServiceId_Length
UpnpSubscriptionRequest.c, 654
UpnpSubscriptionRequest.h, 538
UpnpSubscriptionRequest_get_SID
UpnpSubscriptionRequest.c, 654
UpnpSubscriptionRequest.h, 538
UpnpSubscriptionRequest_get_SID_cstr
UpnpSubscriptionRequest.c, 654
UpnpSubscriptionRequest.h, 538
UpnpSubscriptionRequest_get_SID_Length
UpnpSubscriptionRequest.c, 654
UpnpSubscriptionRequest.h, 539
UpnpSubscriptionRequest_get_UDN
UpnpSubscriptionRequest.c, 655
UpnpSubscriptionRequest.h, 539
UpnpSubscriptionRequest_get_UDN_cstr
UpnpSubscriptionRequest.c, 655
UpnpSubscriptionRequest.h, 539
UpnpSubscriptionRequest_get_UDN_Length
UpnpSubscriptionRequest.c, 655
UpnpSubscriptionRequest.h, 539
UpnpSubscriptionRequest_new
UpnpSubscriptionRequest.c, 655
UpnpSubscriptionRequest.h, 539
UpnpSubscriptionRequest_set_ServiceId
UpnpSubscriptionRequest.c, 655
UpnpSubscriptionRequest.h, 540
UpnpSubscriptionRequest_set_SID
UpnpSubscriptionRequest.c, 656
UpnpSubscriptionRequest.h, 540
UpnpSubscriptionRequest_set_UDN
UpnpSubscriptionRequest.c, 656
UpnpSubscriptionRequest.h, 540
UpnpSubscriptionRequest_strcpy_ServiceId
UpnpSubscriptionRequest.c, 656
UpnpSubscriptionRequest.h, 540
UpnpSubscriptionRequest_strcpy_SID
UpnpSubscriptionRequest.c, 656
UpnpSubscriptionRequest.h, 541
UpnpSubscriptionRequest_strcpy_UDN
UpnpSubscriptionRequest.c, 657
UpnpSubscriptionRequest.h, 541
UpnpSubscriptionRequest_strncpy_ServiceId
UpnpSubscriptionRequest.c, 657
UpnpSubscriptionRequest.h, 541
UpnpSubscriptionRequest_strncpy_SID
UpnpSubscriptionRequest.c, 657
UpnpSubscriptionRequest.h, 541
UpnpSubscriptionRequest_strncpy_UDN
UpnpSubscriptionRequest.c, 657
UpnpSubscriptionRequest.h, 542
UpnpThreadDistribution
UPnP API, 84
upnptools.c

addToAction, 659
ErrorMessages, 660
HEADER_LENGTH, 659
makeAction, 660
UpnpUnRegisterClient
UPnP API, 84
UpnpUnRegisterRootDevice
UPnP API, 84
UpnpUnRegisterRootDeviceLowPower
UPnP API, 85
UpnpUnSubscribe
UPnP API, 86
UpnpUnSubscribeAsync
UPnP API, 87
upnputil.h
linecopy, 829
linecopylen, 829
namecopy, 829
UpnpVirtualDir_set_CloseCallback
UPnP API, 88
UpnpVirtualDir_set_GetInfoCallback
UPnP API, 88
UpnpVirtualDir_set_OpenCallback
UPnP API, 88
UpnpVirtualDir_set_ReadCallback
UPnP API, 88
UpnpVirtualDir_set_SeekCallback
UPnP API, 89
UpnpVirtualDir_set_WriteCallback
UPnP API, 89
UpnpWebFileHandle
UPnP API, 33
UpnpWriteHttpPost
UPnP API, 89
UpnpWriteHttpRequest
UPnP API, 90
URI, 292
uri
http_message_t, 245
uri.c
copy_token, 721
copy_URL_list, 721
free_URL_list, 721
is_escaped, 722
is_mark, 722
is_reserved, 723
is_unreserved, 723
parse_hostport, 723
parse_scheme, 725
parse_uri, 725
parse_uric, 726
remove_dots, 726
remove_escaped_chars, 727
replace_escaped, 727
resolve_rel_url, 728
token_cmp, 728
token_string_casecmp, 729
uri.h
copy_URL_list, 833
free_URL_list, 834
hostport_type, 833
parse_token, 834
parse_uri, 834
print_token, 835
print_uri, 835
remove_dots, 835
remove_escaped_chars, 836
replace_escaped, 837
RESERVED, 833
resolve_rel_url, 837
token, 833
token_cmp, 838
token_string_casecmp, 838
uri_type, 833
URL_list, 833
uri_type
uri.h, 833
URL_LIST, 293
URLs, 294
URL_list
uri.h, 833
urlbuf
http_message_t, 245
URLs
URL_LIST, 294
util.c
linecopy, 730
linecopylen, 730
namecopy, 731
uuid.c
CHECK, 930
format_uuid_v1, 930
format_uuid_v3, 930
get_current_time, 930
read_state, 931
true_random, 931
upnp_uuid_unpack, 931
uuid_compare, 932
uuid_create, 932
uuid_create_from_name, 932
write_state, 933
uuid_compare
uuid.c, 932
uuid_create
uuid.c, 932
uuid_create_from_name
uuid.c, 932
uuid_node_t, 294
uuid_state, 294
cs, 295
node, 295
ts, 295
UUIDS_PER_TICK
sysdep.h, 820

valid_ssdp_msg
SSDP Library, 175

valid_ssdp_notify_hack
http_parser_t, 247
value
http_header_t, 242
VariableCount
TvService, 289
VariableName
TvService, 289
VariableStrVal
TvService, 289
VDCallback_Close
UPnP API, 33
VDCallback_GetInfo
UPnP API, 34
VDCallback_Open
UPnP API, 34
VDCallback_Read
UPnP API, 34
VDCallback_Seek
UPnP API, 35
VDCallback_Write
UPnP API, 35
virtual_Dir_List, 295
virtualDirCallback
UPnP API, 97
VirtualDirCallbacks, 296
close, 297
get_info, 297
open, 297
read, 297
seek, 297
write, 298

web_server_callback
webserver.c, 715
web_server_destroy
webserver.c, 715
web_server_init
webserver.c, 715
web_server_set_alias
webserver.c, 716
web_server_set_cors
webserver.c, 716
web_server_set_root_dir
webserver.c, 717
webserver.c
alias_grab, 706
alias_release, 706
CheckOtherHTTPHeaders, 707
CreateHTTPRangeResponseHeader, 707
ExtraHTTPHeaders, 708
FreeExtraHTTPHeaders, 708
gAliasDoc, 717
get_alias, 708
get_content_type, 709
get_file_info, 709
GetNextRange, 710
glob_alias_init, 710
gMediaTypeList, 717
gMediaTypes, 717
http_RecvPostMessage, 710
is_valid_alias, 711
isFileInVirtualDir, 712
media_list_init, 712
process_request, 712
resp_type, 706
search_extension, 713
StrStr, 714
StrTok, 714
ToUpperCase, 714
web_server_callback, 715
web_server_destroy, 715
web_server_init, 715
web_server_set_alias, 716
web_server_set_cors, 716
web_server_set_root_dir, 717
WinsockInit
UPnP API, 91
WorkerThread
ThreadPool.c, 897
write
VirtualDirCallbacks, 298
write_state
uuid.c, 933

XML API, 176
DOMString, 183
IXML_Attr, 183
IXML_CDATASection, 183
IXML_Document, 184
IXML_Element, 184
IXML_ERRORCODE, 185
IXML_NamedNodeMap, 184
IXML_Node, 184
IXML_NODE_TYPE, 185
IXML_NodeList, 184
IXML_Text, 184
ixmlAttr_free, 185
ixmlCDATASection_free, 185
ixmlCDATASection_init, 185
ixmlCloneDOMString, 186
ixmlDocument_createAttribute, 186
ixmlDocument_createAttributeEx, 187
ixmlDocument_createAttributeNS, 187
ixmlDocument_createAttributeNSEx, 188
ixmlDocument_createCDATASection, 188
ixmlDocument_createCDATASectionEx, 189
ixmlDocument_createDocument, 189
ixmlDocument_createDocumentEx, 189
ixmlDocument_createElement, 190
ixmlDocument_createElementEx, 190
ixmlDocument_createElementNS, 191
ixmlDocument_createElementNSEx, 191
ixmlDocument_createTextNode, 192
ixmlDocument_createTextNodeEx, 192
ixmlDocument_free, 193
ixmlDocument_getElementById, 193
ixmlDocument_getElementsByTagName, 193

ixmlDocument_getElementsByTagNameNS, 194
ixmlDocument_importNode, 194
ixmlDocument_init, 195
ixmlDocumenttoString, 195
ixmlElement_free, 196
ixmlElement_getAttribute, 196
ixmlElement_getAttributeNode, 196
ixmlElement_getAttributeNodeNS, 197
ixmlElement_getAttributeNS, 197
ixmlElement_getElementsByTagName, 198
ixmlElement_getElementsByTagNameNS, 198
ixmlElement_getTagName, 198
ixmlElement_hasAttribute, 199
ixmlElement_hasAttributeNS, 199
ixmlElement_init, 199
ixmlElement_removeAttribute, 200
ixmlElement_removeAttributeNode, 200
ixmlElement_removeAttributeNS, 201
ixmlElement_setAttribute, 201
ixmlElement_setAttributeNode, 202
ixmlElement_setAttributeNodeNS, 202
ixmlElement_setAttributeNS, 203
ixmlFreeDOMString, 204
ixmlLoadDocument, 204
ixmlLoadDocumentEx, 204
ixmlNamedNodeMap_free, 205
ixmlNamedNodeMap_getLength, 205
ixmlNamedNodeMap_getNamedItem, 205
ixmlNamedNodeMap_getNamedItemNS, 206
ixmlNamedNodeMap_item, 206
ixmlNamedNodeMap_removeNamedItem, 206
ixmlNamedNodeMap_removeNamedItemNS, 207
ixmlNamedNodeMap_setNamedItem, 207
ixmlNamedNodeMap_setNamedItemNS, 208
ixmlNode_appendChild, 208
ixmlNode_cloneNode, 209
ixmlNode_free, 209
ixmlNode_getAttributes, 209
ixmlNode_getChildNodes, 210
ixmlNode_getFirstChild, 210
ixmlNode_getLastChild, 211
ixmlNode_getLocalName, 211
ixmlNode_getNamespaceURI, 211
ixmlNode_getNextSibling, 212
ixmlNode_getNodeName, 212
ixmlNode_getNodeType, 213
ixmlNode_getNodeValue, 213
ixmlNode_getOwnerDocument, 214
ixmlNode_getParentNode, 214
ixmlNode_getPrefix, 214
ixmlNode_getPreviousSibling, 215
ixmlNode_hasAttributes, 215
ixmlNode_hasChildNodes, 215
ixmlNode_insertBefore, 216
ixmlNode_removeChild, 216
ixmlNode_replaceChild, 217
ixmlNode_setNodeValue, 218
ixmlNodeList_free, 218
ixmlNodeList_item, 219
ixmlNodeList_length, 219
ixmlNodetoString, 219
ixmlParseBuffer, 220
ixmlParseBufferEx, 220
ixmlPrintDocument, 221
ixmlPrintNode, 221
ixmlRelaxParser, 222
xml_alias_t, 298
ct, 299
doc, 299
last_modified, 299
name, 299
XML_VERSION
gena.h, 740