EBS UPNP DEVICE SDK

Contents

 Embedded UPnP Library version 1.0

1 API Reference

Names

1.1 int **UPnP_RuntimeInit** (UPnPRuntime* rt, UPNP_UINT8* serverAddr, UPNP_UINT16 serverPort, UPNP_INT16 ipType, UPNP_CHAR* wwwRootDir, UPNP_INT16 maxConnections, UPNP_INT16 maxHelperThreads)
Initialize a UPnPRuntime ..... **??**

1.2 void **UPnP_RuntimeDestroy** (UPnPRuntime* rt)
Destroy a UPnPRuntime ..... **??**

1.3 int **UPnP_AddVirtualFile** (UPnPRuntime* rt, const UPNP_CHAR* serverPath, const UPNP_UINT8* data, UPNP_INT32 size, const UPNP_CHAR* contentType)
Create a virtual file on the HTTP server. ..... **??**

1.4 int **UPnP_RemoveVirtualFile** (UPnPRuntime* rt, const UPNP_CHAR* serverPath)
Remove a virtual file from the server ..... **??**

1.5 int **UPnP_ProcessState** (UPnPRuntime* rt, UPNP_INT32 msecTimeout)
Process asynchronous operations in non-threaded mode. ..... **??**

1.6 int **UPnP_StartDaemon** (UPnPRuntime* rt)
Start the UPnP Daemon thread.

??

1.7 int **UPnP_StopDaemon** (UPnPRuntime* rt, UPNP_INT32 secTimeout)

Kill the UPnP Daemon thread. **??**

1.8 const UPNP_CHAR*
UPnP_GetPropertyValueByName (IXML_Document* propertySet, const UPNP_CHAR* name)
Get the value of a named property in a GENA notify message. ... **??**

1.9 const UPNP_CHAR*
UPnP_GetPropertyNameByIndex (IXML_Document* propertySet, int index)
Get the name of the nth property. ..... **??**

1.10 const UPNP_CHAR*
UPnP_GetPropertyValueByIndex (IXML_Document* propertySet, int index)
Get the value of the nth property. ..... **??**

1.11 int **UPnP_AddToPropertySet** (IXML_Document** doc, const UPNP_CHAR* name, const UPNP_CHAR* value)
Add name and value pair to GENA notify message property set. ..... **??**

1.12 int **UPnP_CreateActionResponse** (UPnPActionRequest* request)
Creates a SOAP action response message. ..... **??**

1.13 IXML_Document*

UPnP_CreateAction (const UPNP_CHAR* serviceTypeURI, const UPNP_CHAR* actionName)
Create a SOAP action request. **??**

1.14 int **UPnP_SetActionArg** (IXML_Document* actionDoc, const UPNP_CHAR* name, const UPNP_CHAR* value)
Set an argument for a SOAP action response/request ..... **??**

1.15 int **UPnP_DeviceInit** (UPnPDeviceRuntime* deviceRuntime, UPnPRuntime* rt)
Initialize a UPnPDeviceRuntime
??

1.16 void **UPnP_DeviceFinish** (UPnPDeviceRuntime* deviceRuntime)
Destroy a UPnPDeviceRuntime **??**

1.17 int **UPnP_RegisterRootDevice** (UPnPDeviceRuntime* deviceRuntime, const UPNP_CHAR* descDocURL, IXML_Document* description, UPNP_BOOL autoAddr, UPnPDeviceCallback callback, void* userData, UPnPRootDeviceHandle* retHandle, UPNP_BOOL deviceAdvertise)
Configures the root device and its serives for UPnP ..... **??**

1.18 int **UPnP_UnRegisterRootDevice** (UPnPRootDeviceHandle rootDevice)

Free root device from its server bindings ..... **??**

1.19 int **UPnP_DeviceAdvertise** (UPnPRootDeviceHandle rootDevice, UPNP_INT32 frequencySec, UPNP_INT32 remoteTimeoutSec)

set up the device to send periodic SSDP announcements ..... **??**

1.20 int **UPnP_DeviceNotify** (UPnPDeviceRuntime* deviceRuntime, UPnPRootDeviceHandle rootDevice, const UPNP_CHAR* deviceUDN, const UPNP_CHAR* serviceId, IXML_Document* propertySet)

Sends an event notification message to all the subscribers of the service. ..... **??**

1.21 int **UPnP_DeviceNotifyAsync** (UPnPDeviceRuntime* deviceRuntime, UPnPRootDeviceHandle rootDevice, const UPNP_CHAR* deviceUDN, const UPNP_CHAR* serviceId, IXML_Document* propertySet)

Sends a non blocking event notification message to all the subscribers of the service. ..... **??**

1.22 int **UPnP_AcceptSubscription** (UPnPSubscriptionRequest* subReq, const GENA_CHAR* subscriptionId, UPNP_INT32 timeoutSec, IXML_Document* propertySet, UPNP_INT32 firstNotifyDelayMsec)

Accept a new subscription request. ..... **??**

1.23 int **UPnP_AcceptSubscriptionAsync** (UPnPSubscriptionRequest* subReq, const GENA_CHAR* subscriptionId, UPNP_INT32 timeoutSec, IXML_Document* propertySet, UPNP_INT32 firstNotifyDelayMsec)

Send Subscription Accept asynchronously. ..... **??**

1.24 const UPNP_CHAR* **UPnP_GetRequestedDeviceName** (void* eventStruct, enum e_UPnPDeviceEventType eventType)

Extracts Unique device name for the target device from control/subscription request ..... **??**

1.25 const UPNP_CHAR*

UPnP_GetRequestedServiceId (void* eventStruct, enum e_UPnPDeviceEventType eventType)

Extracts service identifier from a control/subscription request .... **??**

1.26 const UPNP_CHAR*

UPnP_GetRequestedActionName (void* eventStruct, enum e_UPnPDeviceEventType eventType)

Extracts name of the target action from action/subscription request **??**

1.27 void **UPnP_SetActionErrorResponse** (UPnPAction-Request* request, UPNP_CHAR* description, UPNP_INT32 value)

Sets error code and error description for a response to an action request ..... **??**

1.28 const UPNP_CHAR*

UPnP_GetArgValue (UPnPActionRequest* request, const UPNP_CHAR* argName)

Extracts the value of a given argument from an action. ..... **??**

1.29 int **UPnP_SetActionResponseArg** (UPnPActionRequest* request, const UPNP_CHAR* name, const UPNP_CHAR* value)

Inserts name and value of an argument to an action response message ..... **??**

1.1

int **UPnP_RuntimeInit** (UPnPRuntime* rt,
UPNP_UINT8* serverAddr,
UPNP_UINT16 serverPort,
UPNP_INT16 ipType,
UPNP_CHAR* wwwRootDir,
UPNP_INT16 maxConnections,
UPNP_INT16 maxHelperThreads
)

Initialize a UPnPRuntime

Initializes the given UPnPRuntime struct, and sets up an HTTP server instance to receive control/event messages. This function must be called before any other function in the UPnP SDK.

Return Value:	`error` code	
Parameters:	`rt`	pointer to uninitialized UPnPRuntime struct
	`serverAddr`	ip address to bind HTTP server to (NULL for IP_ADDR_ANY)
	`serverPort`	port to bind HTTP server to (0 for ANY_PORT)
	`ipType`	type of ip version used (ipv4 or ipv6), (RTP_NET_TYPE_IPV4 for v4 and RTP_NET_TYPE_IPV6 for v6)
	`wwwRootDir`	HTTP root dir on local file system
	`maxConnections`	the maximum limit on simultaneous HTTP server connections
	`maxHelperThreads`	if UPNP_MULTITHREAD is defined, the max number of helper threads to spawn

1.2

void **UPnP_RuntimeDestroy** (UPnPRuntime* rt)

Destroy a UPnPRuntime

Must be called after all other UPnP SDK calls to clean up runtime data for UPnP.

Return Value: `error` code
Parameters: `rt` pointer to UPnPRuntime struct

1.3

int **UPnP_AddVirtualFile** (UPnPRuntime* rt, const UPNP_CHAR* serverPath, const UPNP_UINT8* data, UPNP_INT32 size, const UPNP_CHAR* contentType)

Create a virtual file on the HTTP server.

Makes the data buffer passed in available at the given path on the HTTP server.

Return Value: `error` code
See Also: UPnP_RemoveVirtualFile

1.4

int **UPnP_RemoveVirtualFile** (UPnPRuntime* rt, const UPNP_CHAR* serverPath)

Remove a virtual file from the server

Must be called before UPnP_RuntimeDestroy to remove any virtual files added using UPnP_AddVirtualFile.

Return Value: `error` code

1.5

int **UPnP_ProcessState** (UPnPRuntime* rt,
UPNP_INT32 msecTimeout
)

Process asynchronous operations in non-threaded mode.

This function blocks for at most msecTimeout milliseconds, processing any asynchronous operations that may be in progress on either the control point or device runtime attached to the given UPnPRuntime.

This function must be called in order to receive events if an application is running with the UPnP SDK in single-threaded mode.

Return Value: `error` code

Parameters:

`rt`	pointer to UPnPRuntime struct
`msecTimeout`	time in miliseconds for which the function blocks

1.6

int **UPnP_StartDaemon** (UPnPRuntime* rt)

Start the UPnP Daemon thread.

This function must be called in multithreaded mode to start the UPnP daemon, which listens for requests/announcements on the network, and sends any events to the attached control point/device runtime.

Return Value: `error` code
Parameters: `rt` pointer to UPnPRuntime struct
See Also: UPnP_StopDaemon

1.7

int **UPnP_StopDaemon** (UPnPRuntime* rt, UPNP_INT32 secTimeout)

Kill the UPnP Daemon thread.

This function stops the UPnP daemon from executing. It will wait for at most secTimeout seconds for all helper threads to terminate. If this function returns negative error code, it means the timeout expired without the successful termination of one or more helper threads. In this case, calling UPnP_RuntimeDestroy may cause a fault since there are still helper threads running that may try to access the data structures pointed to by the UPnPRuntime.

Return Value: `error` code
Parameters:

`rt`	the device runtime to stop
`secTimeout`	time to wait for daemon to stop

1.8

const UPNP_CHAR* **UPnP_GetPropertyValueByName** (IXML_Document* propertySet, const UPNP_CHAR* name)

Get the value of a named property in a GENA notify message.

The string returned must not be modified in any way. It is valid until the IXML_Document is deleted.

Return Value: `the` value or NULL if the property was not found

1.9

const UPNP_CHAR* **UPnP_GetPropertyNameByIndex** (IXML_Document* propertySet, int index)

Get the name of the nth property.

The string returned must not be modified in any way. It is valid until the IXML_Document is deleted.

Return Value: `the` value or NULL if the property was not found

Parameters: `propertySet` address of xml property set
`index` index in property for value

1.10

const UPNP_CHAR* **UPnP_GetPropertyValueByIndex** (IXML_Document* propertySet, int index)

Get the value of the nth property.

The string returned must not be modified in any way. It is valid until the IXML_Document is deleted.

Return Value: `the` value or NULL if the property was not found

Parameters: `propertySet` address of xml property set
`index` index in property for value

1.11

int **UPnP_AddToPropertySet** (IXML_Document** doc, const UPNP_CHAR* name, const UPNP_CHAR* value)

Add name and value pair to GENA notify message property set.

Add a new name value pair entry to the property set

Return Value:	`error`	code
Parameters:	`doc`	address of property set
	`name`	pointer to name for new entry
	`value`	address of value of for the new entry

1.12

int **UPnP_CreateActionResponse** (UPnPActionRequest* request)

Creates a SOAP action response message.

Creates a response message skeleton for the supplied SOAP action request

Return Value:	`error`	code

1.13

IXML_Document* **UPnP_CreateAction** (const UPNP_CHAR* serviceTypeURI, const UPNP_CHAR* actionName)

Create a SOAP action request.

Creates an XML document which will hold the SOAP action request message. This function returns the address of newly formed XML document. After finishing the process of sending action request the application must release this xml document.

Return Value:	`pointer`	to newly created IXML_Document, which can be passed intoUPnP_SetActionArg to set the action arguments; NULL on error
Parameters:	`serviceTypeURI`	string containing service type of the target service
	`actionName`	name on action on the target service

1.14

int **UPnP_SetActionArg** (IXML_Document* actionDoc, const UPNP_CHAR* name, const UPNP_CHAR* value)

Set an argument for a SOAP action response/request

This function can be used on an IXML_Document created by either UPnP_CreateActionResponse (→ 1.13, *page* **??**) or UPnP_CreateAction to set either the input or output arguments for a SOAP action.

Return Value:	`error`	code

Parameters:	`actionDoc`	pointer to action respose message
	`name`	argument name *
	`value`	argument value *

1.15

int **UPnP_DeviceInit** (UPnPDeviceRuntime* deviceRuntime, UPnPRuntime* rt)

Initialize a UPnPDeviceRuntime

Initializes all device state data in a UPnPDeviceRuntime struct (allocated by the calling application), and binds the device to the specified UPnPRuntime. The UPnPRuntime must be initialized via UPnP_RuntimeInit before this function is called. Only one device may be bound to a single UPnPRuntime at once. This function must be called before all other device related functions.

Return Value:	`error`	code
Parameters:	`deviceRuntime`	pointer to the device runtime buffer
	`rt`	pointer to an initialized upnp runtime-buffer.
See Also:	UPnP_DeviceFinish	

1.16

void **UPnP_DeviceFinish** (UPnPDeviceRuntime* deviceRuntime)

Destroy a UPnPDeviceRuntime

Cleans up all data associated with a UPnPDeviceRuntime structure. Once this function has been called, it is safe to free the memory used by the UPnPDeviceRuntime structure.

Return Value:	`error`	code
Parameters:	`deviceRuntime`	address of runtime of device to destroy

See Also: UPnP_DeviceInit

1.17

int **UPnP_RegisterRootDevice** (UPnPDeviceRuntime* deviceRuntime, const UPNP_CHAR* descDocURL, IXML_Document* description, UPNP_BOOL autoAddr, UPnPDeviceCallback callback, void* userData, UPnPRootDeviceHandle* retHandle, UPNP_BOOL deviceAdvertise)

Configures the root device and its serives for UPnP

Sets up the device to serve UPnP requests from the clients; set up devcive for ssdp announcements if deviceAdvertise is turned on.

Return Value: `error` code

Parameters:

`deviceRuntime`	device runtime information
`descDocURL`	relative url of devicedescription document
`description`	address of DOM representationof the device descriptiondocument
`autoAddr`	Select swtich for Auto IPif 1 - uses AutoIPif 0 - extracts address from thedevice description document
`callback`	pointer to the callback functionfor the device
`userData`	user data for callback
`retHandle`	handle to the current root device
`deviceAdvertise`	Switch to turn ON and OFF device advertisingIf 1 - device will be set up to send periodicSSDP announcements.If 0 - no ssdp announcements will be send

1.18

int **UPnP_UnRegisterRootDevice** (UPnPRootDeviceHandle rootDevice)

Free root device from its server bindings

Unregisters the root device from the internal server, so that the future UPnP requests will not be served for this root device.

Return Value: `error` code
Parameters: `rootDevice` Handle to root device

1.19

int **UPnP_DeviceAdvertise** (UPnPRootDeviceHandle rootDevice, UPNP_INT32 frequencySec, UPNP_INT32 remoteTimeoutSec)

set up the device to send periodic SSDP announcements

The function prepares the device to send periodic announcements every frequecySec seconds.

Return Value:	`error` code	
Parameters:	`rootDevice`	handle to the device
	`frequencySec`	interval in secondsbetween two announcements
	`remoteTimeoutSec`	time in seconds for which the remote-client will cache the information inthe announcement

1.20

int **UPnP_DeviceNotify** (UPnPDeviceRuntime* deviceRuntime, UPnPRootDeviceHandle rootDevice, const UPNP_CHAR* deviceUDN, const UPNP_CHAR* serviceId, IXML_Document* propertySet)

Sends an event notification message to all the subscribers of the service.

Sends an event notification message to all the subscribers of the service.

Return Value:	`error` code	
Parameters:	`deviceRuntime`	device runtime information
	`rootDevice`	handle to the device
	`deviceUDN`	unique device identifier (UUID in thedevice description document) forthe device
	`serviceId`	unique service identifier (serviceIDin the device description document)for the service
	`propertySet`	contains the evented variable andits value in XML format.

1.21

int **UPnP_DeviceNotifyAsync** (UPnPDeviceRuntime* deviceRuntime, UPnPRootDeviceHandle rootDevice, const UPNP_CHAR* deviceUDN, const UPNP_CHAR* serviceId, IXML_Document* propertySet)

Sends a non blocking event notification message to all the subscribers of the service.

Sends a non blocking event notification message to all the subscribers of the service.

Return Value:	`error`	code
Parameters:	`deviceRuntime`	the runtime for the device
	`rootDevice`	the root device
	`deviceUDN`	the UDN of the specific device
	`serviceId`	the ID of the service notifying
	`propertySet`	property set to send as new values

1.22

int **UPnP_AcceptSubscription** (UPnPSubscriptionRequest* subReq, const GENA_CHAR* subscriptionId, UPNP_INT32 timeoutSec, IXML_Document* propertySet, UPNP_INT32 firstNotifyDelayMsec)

Accept a new subscription request.

This function adds a new subscriber device's internal subscriber's list, generates a unique subscription Id for this subscriber, sets a duration in seconds for this subscription to be valid and sends a subscription response indicating success or failure to subscription request.

Return Value:	`error`	code

Parameters:	`subReq`	address of structure containing subscriptionrequest information
	`subscriptionId`	subscription identifier for the subscriber
	`timeoutSec`	duration in seconds for which the subscriptionis valid
	`propertySet`	pointer to response message in XML format.
	`firstNotifyDelayMsec`	delay in milliseconds before sending thefirst event notification to the new subscriber

1.23

int **UPnP_AcceptSubscriptionAsync** (UPnPSubscriptionRequest* subReq, const GENA_CHAR* subscriptionId, UPNP_INT32 timeoutSec, IXML_Document* propertySet, UPNP_INT32 firstNotifyDelayMsec)

Send Subscription Accept asynchronously.

Send Subscription Accept asynchronously. Optional parameters may be given a value of zero to indicate use default.

Return Value:	`error` code	
Parameters:	`subReq`	the request being accepted
	`subscriptionId`	alternate subscription ID (optional)
	`timeoutSec`	remote cache timeout value inseconds (optional)
	`propertySet`	property set for initial notify
	`firstNotifyDelayMsec`	delay in milliseconds before sending thefirst event notification to the new subscriber

1.24

const UPNP_CHAR* **UPnP_GetRequestedDeviceName** (void* eventStruct, enum e_UPnPDeviceEventType eventType)

Extracts Unique device name for the target device from control/subscription request

Extracts Unique device name for the target device from control/subscription request

Return Value:	`error`	code
Parameters:	`eventStruct`	pointer to request structure
	`eventType`	type of event

1.25

const UPNP_CHAR* **UPnP_GetRequestedServiceId** (void* eventStruct, enum e_UPnPDeviceEventType eventType)

Extracts service identifier from a control/subscription request

Extracts service identifier from a control/subscription request

Return Value:	`error`	code
Parameters:	`eventStruct`	pointer to request structure
	`eventType`	type of event

1.26

const UPNP_CHAR* **UPnP_GetRequestedActionName** (void* eventStruct, enum e_UPnPDeviceEventType eventType)

Extracts name of the target action from action/subscription request

Extracts name of the target action from action/subscription request

Return Value: `pointer` to action name is available
Parameters: `eventStruct` pointer to request structure
`eventType` type of event

1.27

void **UPnP_SetActionErrorResponse** (UPnPActionRequest* request, UPNP_CHAR* description, UPNP_INT32 value)

Sets error code and error description for a response to an action request

Sets error code and error description for a response to an action request

Return Value: `none`

1.28

const UPNP_CHAR* **UPnP_GetArgValue** (UPnPActionRequest* request, const UPNP_CHAR* argName)

Extracts the value of a given argument from an action.

Extracts the value of a given argument from an action. Action information is stored in form of IXML element.

Return Value: `error` code

1.29

int **UPnP_SetActionResponseArg** (UPnPActionRequest* request, const UPNP_CHAR* name, const UPNP_CHAR* value)

Inserts name and value of an argument to an action response message

Inserts name and value of an argument to an action response message

Return Value: `error` code

2 Internal Library Documentation

Names

These functions are not at the API level and should not be called from code outside the UPnP library.

2.1 Discovery/SSDP

Names

Reads all messages posted to the multicast address ..... **??**

2.1.10 int **_SSDP_ReadMSearchHeader** (void* request, HTTPSession* ptr, HTTPHeaderType type, const HTTP_CHAR* name, const HTTP_CHAR* value)

Extracts MX and St headers from a SSDP request ..... **??**

2.1.11 int **_SSDP_ReadNotifyHeader** (void* request, HTTPSession* ptr, HTTPHeaderType type, const HTTP_CHAR* name, const HTTP_CHAR* value)

Extracts MX and St headers from a SSDP request ..... **??**

2.1.12 SSDP_INT32 **SSDP_QueueSearchResponse** (SSDPServerContext* ctx, SSDPSearch* search, const SSDP_CHAR* targetLocation, const SSDP_CHAR* targetURN, SSDP_UINT32 targetTimeoutSec)

Queues a response to the response list based on its scheduled delivery time. ..... **??**

2.1.13 SSDP_INT32

SSDP_CheckPendingResponses (SSDPServerContext* ctx, SSDP_UINT32 currentTimeMsec)
Delivers responses scheduled for delivery. ..... **??**

2.1.14 void **SSDP_ProcessError** (SSDP_CHAR* errMsg)
Process SSDP Errors ..... **??**

2.1.15 SSDP_UINT32
SSDP_RandMax (SSDP_UINT32 mxLimit)
generates a random number between 0 and mxLimit ..... **??**

2.1.16 int **UPnP_DeviceSSDPCallback** (SSDPServerContext* ctx, SSDPServerRequest* serverRequest, void* cookie)
SSDP callback for UPnP. ..... **??**

2.1.17 int **UPnP_DeviceSendAllAlive** (UPnPDeviceRuntime* runtime)
Sends alive advertisements for everything under all the root devices. ..... **??**

2.1.18 int **UPnP_DeviceSendRootDeviceAlive** (UPnPRootDevice* rootDevice, int deep)
Sends alive advertisements for a root device. ..... **??**

2.1.19 int **UPnP_DeviceSendDeviceAlive** (UPnPDevice* device)
Sends alive notifications for a device ..... **??**

2.1.20 int **UPnP_DeviceSendServiceAlive** (UPnPService* service)

Sends alive notifications for a service ..... **??**

2.1.21 int **_UPnP_DeviceSendDeviceAlive** (UPnPDevice* device)

Sends alive advertisements for a device. ..... **??**

2.1.22 int **_UPnP_DeviceSendServiceAlive** (UPnPService* service)

Sends alive advertisements for a service. ..... **??**

2.1.23 int **_UPnP_DeviceSendRootDeviceAlive** (UPnPRootDevice* rootDevice, int deep)

Sends alive advertisements for every thing under all root devices. ..... **??**

2.1.24 int **UPnP_DeviceSendAllByeBye** (UPnPDeviceRuntime* runtime)

Sends bye-bye advertisements for each device and associated service for all the root devices. ..... **??**

2.1.25 int **UPnP_DeviceSendRootDeviceByeBye** (UPnPRootDevice* rootDevice, int deep)

Sends bye-bye notifications for a root device ..... **??**

2.1.26 int **UPnP_DeviceSendDeviceByeBye** (UPnPDevice* device)

Sends bye-bye notifications for the device ..... **??**

2.1.27 int **UPnP_DeviceSendServiceByeBye** (UPnPService* service)

These functions are not at the API level and should not be called from code outside the UPnP library.

2.1.1

SSDP_INT32 **SSDP_ServerInit** (SSDPServerContext* ctx, SSDP_UINT8* ipAddr, SSDP_INT16 ipType, const SSDP_CHAR* serverId, SSDPCallback cb, void* cookie)

SSDP server initialization routine.

This routine starts up SSDP services by creating a UDP socket, getting the ssdp multicast membership for the socket and initlializing ssdp context.

Return Value: `error` code

Parameters:

`ctx`	pointer to an uninitialized SSDP context structure
`ipAddr`	IP address of the host to bind the UDP socket to, if a NULL issupplied, the UDP socket is bound to the local IP address
`ipType`	ip version 4 or ipversion 6
`serverId`	String holding platform name
`cb`	SSDP Callback routine
`cookie`	cookie(runtime) to be stored in ssdp context to be passedlater to ssdp callback

2.1.2

SSDP_INT32 **SSDP_ServerAddToSelectList** (SSDPServerContext* ctx, RTP_FD_SET* readList, RTP_FD_SET* writeList, RTP_FD_SET* errList)

2.1.3

SSDP_BOOL **SSDP_ServerProcessState** (SSDPServerContext* ctx, RTP_FD_SET* readList, RTP_FD_SET* writeList, RTP_FD_SET* errList)

2.1.4

void **SSDP_ServerDestroy** (SSDPServerContext* ctx

pointer to SSDP context* /)

2.1.5

SSDP_INT32 **SSDP_SendNotify** (SSDPServerContext* ctx, const SSDP_CHAR* notifyType, const SSDP_CHAR* notifySubType, const SSDP_CHAR* usn, const SSDP_CHAR* location, SSDP_UINT32* timeout)

Send a SSDP notification for the device or service.

This routine sends ssdp alive and bye-bye notifications on the mulicast address for devices and servces

Return Value:	`error`	code
Parameters:	`ctx`	pointer to SSDP context
	`notifyType`	the notification type (NT) string
	`notifySubType`	pointer to string containing NT subtype
	`usn`	pointer to string containing USN header
	`location`	pointer to string containing Location header
	`timeout`	pointer to max-age header value

2.1.6

SSDP_INT32 **SSDP_SendResponse** (SSDPServerContext* ctx, SSDPPendingResponse* response)

Deliver a responce to SSDP discovery request.

Deliver a responce to SSDP discovery request.

Return Value: `error` code
Parameters: `ctx` pointer to SSDP context
`response` buffer holding response information

2.1.7

SSDP_INT32 **_SSDP_ProcessOneRequest** (SSDPServerContext* ctx)

Process an incoming SSDP discovery request.

Processes a SSDP discovery request available during a period given by timeoutMsec milli seconds.

Return Value: `error` code
Parameters: `ctx` pointer to SSDP context

2.1.8

int **SSDP_ParseRequest** (SSDPServerContext* ctx, SSDPServerRequest* ssdpRequest)

Extract SSDP request.

Retrieves messages from the multicast address, if ssdp request is detected a buffer holding request information is populated

Return Value: `error` code
Parameters: `ctx` pointer to SSDP context
`ssdpRequest` address of the SSDPServerEvent structure to fill up

2.1.9

int **SSDP_McastRead** (void* cookie, SSDP_UINT8* buffer, SSDP_INT32 min, SSDP_INT32 max)

Reads all messages posted to the multicast address

Reads all messages posted to the multicast address

Return Value:	`error`	code
Parameters:	`cookie`	internal cookie
	`buffer`	pointer to buffer contaning request message
	`min`	
	`max`	max size to be read

2.1.10

int **_SSDP_ReadMSearchHeader** (void* request, HTTPSession* ptr, HTTPHeaderType type, const HTTP_CHAR* name, const HTTP_CHAR* value)

Extracts MX and St headers from a SSDP request

Extracts MX and St headers from a SSDP request

Return Value:	`error`	code
Parameters:	`request`	request buffer to be populated
	`ptr`	current HTTP session
	`type`	HTTP header type
	`name`	holds the name of the header
	`value`	holds the value of the header

2.1.11

int **_SSDP_ReadNotifyHeader** (void* request, HTTPSession* ptr, HTTPHeaderType type, const HTTP_CHAR* name, const HTTP_CHAR* value)

Extracts MX and St headers from a SSDP request

Extracts MX and St headers from a SSDP request

Return Value: `error` code

Parameters:

`request`	request buffer to be populated
`ptr`	current HTTP session
`type`	HTTP header type
`name`	holds the name of the header
`value`	holds the value of the header

2.1.12

SSDP_INT32 **SSDP_QueueSearchResponse** (SSDPServerContext* ctx, SSDPSearch* search, const SSDP_CHAR* targetLocation, const SSDP_CHAR* targetURN, SSDP_UINT32 targetTimeoutSec)

Queues a response to the response list based on its scheduled delivery time.

Queues a response to the response list based on its scheduled delivery time. A random delivery time within targetTimeoutSec duration is calculated. This response is positioned in the list according to its scheduled delivery time.

Return Value: `error` code

Parameters:	`ctx`	pointer to SSDP context
	`search`	pointer to the buffer containingthe request information
	`targetLocation`	pointer to string containing Location header
	`targetURN`	pointer to string containing USN header
	`targetTimeoutSec`	max-age header value

2.1.13

SSDP_INT32 **SSDP_CheckPendingResponses** (SSDPServerContext* ctx, SSDP_UINT32 currentTimeMsec)

Delivers responses scheduled for delivery.

Scan the pending response list and deliver responses for which the scheduled time count is less than supplied time count.

Return Value:	`error` code	
Parameters:	`ctx`	pointer to SSDP context
	`currentTimeMsec`	time against which the scheduledtime is checked

2.1.14

void **SSDP_ProcessError** (SSDP_CHAR* errMsg)

Process SSDP Errors

Process SSDP Errors

Return Value:	`None`	
Parameters:	`errMsg`	error message string

2.1.15

SSDP_UINT32 **SSDP_RandMax** (SSDP_UINT32 mxLimit)

generates a random number between 0 and mxLimit

generates a random number between 0 and mxLimit

Return Value: `error` code
Parameters: `mxLimit` upper limit of the random number

2.1.16

int **UPnP_DeviceSSDPCallback** (SSDPServerContext* ctx, SSDPServerRequest* serverRequest, void* cookie)

SSDP callback for UPnP.

The callback function checks the type of SSDP request and creates a corresponding response. This response is queued in a list with a scheduled delivery time

Return Value:	`error`	code
Parameters:	`ctx`	the SSDP context
	`serverRequest`	address of buffer holding ssdp requestinformation
	`cookie`	cookie holds pointer to device run timeinformation

2.1.17

int **UPnP_DeviceSendAllAlive** (UPnPDeviceRuntime* runtime)

Sends alive advertisements for everything under all the root devices.

Sends alive advertisements for each device and associated service for all the root devices.

Return Value: `error` code
Parameters: `runtime` address of devices current runtime state

2.1.18

int **UPnP_DeviceSendRootDeviceAlive** (UPnPRoot-Device* rootDevice, int deep)

Sends alive advertisements for a root device.

Sends alive advertisements for a root device.

Return Value:	`error`	code
Parameters:	`rootDevice`	address of the root device
	`deep`	if 1, send advertisements for all embedded dives and services if 0, send advertisements for the device under the root and itsassociated services

2.1.19

int **UPnP_DeviceSendDeviceAlive** (UPnPDevice* device)

Sends alive notifications for a device

Sends alive notifications for a device

Return Value: `error` code
Parameters: `device` pointer to the device

2.1.20

int **UPnP_DeviceSendServiceAlive** (UPnPService* service)

Sends alive notifications for a service

Sends alive notifications for a service

Return Value: `error` code
Parameters: `service` pointer to a service

2.1.21

int **_UPnP_DeviceSendDeviceAlive** (UPnPDevice* device)

Sends alive advertisements for a device.

Sends alive advertisements for a device.

Return Value: `error` code
Parameters: `device` pointer to the device

2.1.22

int **_UPnP_DeviceSendServiceAlive** (UPnPService* service)

Sends alive advertisements for a service.

Sends alive advertisements for a service.

Return Value: `error` code
Parameters: `service` pointer to the service

2.1.23

int **_UPnP_DeviceSendRootDeviceAlive** (UPnPRootDevice* rootDevice, int deep)

Sends alive advertisements for every thing under all root devices.

Sends alive advertisements for each device and associated service for all the root devices.

Return Value: `error` code
Parameters:

`rootDevice`	pointer to the root device
`deep`	if 1, send advertisements for all embedded dives and services if 0, send advertisements for the device under the root and itsassociated services

2.1.24

int **UPnP_DeviceSendAllByeBye** (UPnPDeviceRuntime* runtime)

Sends bye-bye advertisements for each device and associated service for all the root devices.

Sends bye-bye advertisements for each device and associated service for all the root devices.

Return Value: `error` code
Parameters: `runtime` address of devices current runtime state

2.1.25

int **UPnP_DeviceSendRootDeviceByeBye** (UPnPRootDevice* rootDevice, int deep)

Sends bye-bye notifications for a root device

Sends bye-bye notifications for a root device

Return Value:	`error`	code
Parameters:	`rootDevice`	address of the root device
	`deep`	if 1, send advertisements for all embedded dives and services if 0, send advertisements for the device under the root and itsassociated services

2.1.26

int **UPnP_DeviceSendDeviceByeBye** (UPnPDevice* device)

Sends bye-bye notifications for the device

Sends bye-bye notifications for the device

Return Value: `error` code
Parameters: `device` pointer to the device

2.1.27

int **UPnP_DeviceSendServiceByeBye** (UPnPService* service)

Sends bye-bye notifications for the service

Sends bye-bye notifications for the service

Return Value: `error` code
Parameters: `service` pointer to the service

2.1.28

int **_UPnP_DeviceSendRootDeviceByeBye** (UPnPRootDevice* rootDevice, int deep)

Sends bye-bye notifications for root device.

Sends bye-bye notifications for root device.

Return Value:	`error`	code
Parameters:	`rootDevice`	address of the root device
	`deep`	if 1, send advertisements for all embedded devicesand services. if 0, send advertisements for thedevice under the root and its associated services

2.1.29

int **_UPnP_DeviceSendDeviceByeBye** (UPnPDevice* device)

Sends bye-bye notifications for the device

Sends bye-bye notifications for the device

Return Value: `error` code
Parameters: `device` address of the device

2.1.30

int **_UPnP_DeviceSendServiceByeBye** (UPnPService* service)

Sends bye-bye notifications a service.

Sends bye-bye notifications a service.

Return Value: `error` code
Parameters: `service` address of the service

2.2

Description

Names

2.2.1 UPnPRootDevice* **UPnP_DeviceDescribeRootDevice** (IXML_Document* doc, int maxDepth)

Get the root device elements needed by SSDP, GENA and SOAP. ..... **??**

2.2.2 UPnPDevice*

These functions are not at the API level and should not be called from code outside the UPnP library.

2.2.1

UPnPRootDevice* **UPnP_DeviceDescribeRootDevice** (IXML_Document* doc, int maxDepth)

Get the root device elements needed by SSDP, GENA and SOAP.

Get the root device elements needed by SSDP, GENA and SOAP.

Return Value:	`Address`	of UPnPRootDevice buffer containing root device information
Parameters:	`doc`	pointer to device document xml page
	`maxDepth`	indicates the depth level this funtion-will search the dom tree for device-sembedded within this device. If maxDepth= x, where x > 0, then UP-nPDevicebuffer will search for embed-deddevices and related services that are xlevel deep in the XML DOM tree. IfmaxDepth = 0 then only the device and itsrelated service informa-tion are described

2.2.2

UPnPDevice* **UPnP_DeviceDescribeDevice** (IXML_Element* deviceElement, UPnPRootDevice* root-Device, int maxDepth)

Extracts the device information needed by SSDP, GENA and SOAP.

Extracts the device information needed by SSDP, GENA and SOAP.

Return Value:	`Address`	of UPnPDevice buffer having device information
Parameters:	`deviceElement`	location of device information insidexml dom tree
	`rootDevice`	pointer to the device's root device
	`maxDepth`	indicates the depth level this funtionwill search the dom tree for devicesembedded within this device. If maxDepth= x, where x > 0, then UPnPDevicebuffer will search for embeddeddevices and related services that are xlevel deep in the XML DOM tree. IfmaxDepth = 0 then only the device and itsrelated service information are described

2.2.3

void **_UPnP_DeviceDescribeDevice** (UPnPDevice* device, IXML_Element* deviceElement, UPnPRootDevice* rootDevice, int maxDepth)

Extracts device information.

Extracts device information. Gets the required information from a XML document searching maxDepth deep for any embedded devices and related services under this device

Return Value:	`None`	
Parameters:	`device`	pointer to UPnPDevice buffer to storeextracted information in
	`deviceElement`	location of device information insidexml dom tree
	`rootDevice`	pointer to the device's root device
	`maxDepth`	indicates the depth level this funtionwill search the dom tree for devicesembedded within this device

2.2.4

UPnPService* **UPnP_DeviceDescribeService** (IXML_Element* serviceElement, UPnPDevice* device)

Extracts the service information.

Extracts the service information.

Return Value:	`Address`	of UPnPService buffer containing service information
Parameters:	`serviceElement`	location of service information insidexml dom tree
	`device`	address of device this service belongsto

2.2.5

void **_UPnP_DeviceDescribeService** (UPnPService* serviceNode, IXML_Element* serviceElement, UPnPDevice* device)

Extracts service information.

Extracts service information. Gets the service related information from a XML document

Return Value:	`None`	
Parameters:	`serviceNode`	pointer to UPnPService buffer to storeextracted information in
	`serviceElement`	location of service information insidexml dom tree
	`device`	address of device this service belongsto

2.2.6

void **UPnP_DeviceFreeRootDevice** (UPnPRootDevice* rootDevice)

Frees the resources used by a UPnPRootDevice type root device

Frees the resources used by a UPnPRootDevice type root device

Return Value: `None`

Parameters:

`rootDevice`	pointer to root device whose resource-sare to be freed

2.2.7

void **UPnP_DeviceFreeDevice** (UPnPDevice* device)

Frees the resources used by the device

Frees the resources used by the device

Return Value: `None`

Parameters:

`device`	pointer to device whose resources are to be freed

2.2.8

void **UPnP_DeviceFreeService** (UPnPService* service)

Frees the resources used by the device

Frees the resources used by the device

Return Value: `None`

Parameters:

`service`	pointer to service whose resources are to be freed

2.3

Control/SOAP

Names

These functions are not at the API level and should not be called from code outside the UPnP library.

2.3.1

int **UPnP_DeviceControlBindService** (UPnPDeviceRuntime* deviceRuntime, UPnPService* service)

Initialize a service for control.

Initialize a service for control. Binds a service control URL to the HTTP server to enable SOAP action processing for that service.

Return Value:	`error`	code
Parameters:	`deviceRuntime`	the device runtime that owns theservice.
	`service`	the service to bind

2.3.2

int **UPnP_DeviceControlUnBindService** (UPnPDeviceRuntime* deviceRuntime, UPnPService* service)

Stop control on a service.

Stop control on a service. Removes the HTTP server binding for this service's control URL.

Return Value:	`error`	code
Parameters:	`deviceRuntime`	the device runtime that owns theservice.
	`service`	the service to unbind

2.3.3

int **UPnP_DeviceControlSOAPCallback** (SOAPServerContext* ctx, SOAPRequest* request, void* cookie)

SOAP callback for UPnP.

SOAP callback for UPnP. Handles a single incoming SOAP control request by parsing the action document and calling the device callback.

Return Value:	`error`	code; tells SOAP whether the action was successful orgenerated a fault
Parameters:	`ctx`	the SOAP context
	`request`	the action request
	`cookie`	the UPnPService being controlled

2.3.4

char* **_getControlURL** (IXML_Document* descDoc, IXML_Element* serviceElement)

Calculates server-relative control URL for a service.

Calculates server-relative control URL for a service. For example, if the service's absolute control URL is "http://192.168.1.110:5423/myService.0001", this function will allocate and return the string: "/myService.0001".

Return Value:	`error`	code
Parameters:	`descDoc`	the service's device description document
	`serviceElement`	the service element inside descDoc

2.4

Eventing/GENA

These functions are not at the API level and should not be called from code outside the UPnP library.

3 Examples

Names

3.1 UPnP Device Initialization Example

Setting up a UPnP Device

This code demonstrates in brief the necessary steps to set up a UPnP device for discovery, description, control, and eventing.

<pre> int main (void) int result; IXML_Document *xmlDevice; UPnPRuntime rt; UPnPRootDeviceHandle rootDevice;

// UPnP maintains a runtime structure; The first step is to // initialize UPnPRuntime struct. UPnP_RuntimeInit takes a // pointer to an uninitialized UPnPRuntime struct and other // necessary necessary data to initialize and populate upnp // the engine. result = UPnP_RuntimeInit (&rt, 0, // serverAddr: IP_ANY_ADDR 0, // serverPort: any port RTP_NET_TYPE_IPV4, // ipv4 "c:
www-root
", // web server root dir 10, // maxConnections 5 // maxHelperThreads);

if (result < 0) return (-1);

// Next, we need a UPnPDeviceRuntime; UPnP_DeviceInit takes // a pointer to an uninitialized UPnPDeviceRuntime struct // and does all necessary initialization.

result = UPnP_DeviceInit (&deviceRuntime, &rt);

if (result < 0) return (-1);

// Load the root device description page into memory. xmlDevice = ixmlLoadDocument("c:

www-root
device.xml"); if (!xmlDevice) return (-1);

result = UPnP_RegisterRootDevice (&deviceRuntime, "device.xml", xmlDevice, 1, // auto address resolution testDeviceCallback, 0, // userData for callback &rootDevice, 1 // advertise);

if (result < 0) return (-1);

UPnP_DeviceAdvertise(rootDevice, ANNOUNCE_FREQUENCY_SEC, REMOTE_CACHE_TIMEOUT_SEC);

// start the UPnP daemon thread UPnP_StartDaemon(&rt);

// for polled mode, use this

//while (1) // //UPnP_ProcessState (&rt,1000); //printf("."); //

</pre>