# Twonky Server REST API Specification

Download as PDF



## Introduction

An application using the REST APIs can implement any kind of home network audio, video or image control functionality, from a generic AV controller application to a dedicated picture slideshow controller that allows watching photos on your TV set, controlled from a mobile phone. These APIs allow controlling the Twonky DLNA stack through a set of REST calls and retrieve metadata in RSS or JSON format.

The Twonky DLNA stack implements the UPnP AV Control Point (CP) and DLNA Digital Media Controller (DMC). This includes the media server control point (MSCP) and media renderer control point (MRCP) functionality as well as DLNA upload and download controller. This functionality allows to control UPnP and DLNA based home network media servers like Twonky Server and Microsoft Windows Media Player Sharing as well as any UPnP AV and DLNA based media renderer devices.

This document is the technical specification of these REST APIs. It assumes the reader is familiar with related web technologies and standards, as well as with the basics of UPnP and DLNA home network technology. Some features of the REST APIs are only available in case the client is integrated with a DTCP-IP enabled Twonky Server.

## Use Cases

The REST APIs enable the implementation of a DLNA control point and DLNA player without the need for a DLNA stack. The API supports discovery and controlling of media servers and media renderers in the home network, as well as the DLNA upload (UP) and download (DN). The media server control API supports browse, search and sort, as well as storage and retrieval of server side WPL playlists. The DLNA push controller feature allows selecting media items on a media server, adding them to a play queue and sending them to DLNA renderers in the home network.

## Concepts

Media content on a media server can come with a huge list of associated metadata. To easily reference a specific media content item on a media server (or in a play queue) the client uses bookmarks. **Bookmarks** are the recommended way to reference and exchange position information and media items within the application. Bookmarks are NOT persistent and only designed for short-term use within the APIs. So an application can request a bookmark for a specific media item on a media server and tell the client to add this item to a play queue without the need to handle any actual metadata for this item. Bookmarks are basically strings but are opaque and shall not be manipulated by applications in any form.

**All API calls are blocking calls**. They only return after the request was completed. For all calls involving network access this can take relatively long time, several seconds are possible for a busy media server. Worst case is a time out (e.g. the target device is dead), taking up to a minute. Depending on the type of application this can create serious issues. If it's a headless application performing some background action, it might be irrelevant. For all UI-based applications this means most calls to the APIs need to be decoupled from the UI threads. A typical implementation would provide a request queue with callbacks on completion of the call. The application would implement some state machine on top of this, for example to show a spinning wheel while the request is being processed and allow to simply aborting it. Or in a UI automatically abort the request if the user selects to do something else.

By using the API a **play queue** for each media renderer in the network is automatically created. By using the bookmarks an application can add media items to a queue in one simple step without the need to handle a set of metadata. It's also possible to add media items to a play queue by setting all required metadata directly.

## Control point design considerations

The REST APIs provide a powerful interface to browse and search media servers, and control media renderers in a home network and provide a robust way to render playlists on media renderers automatically. It abstracts much of the complexity of network access and network device control from the application development. Nonetheless the controlled devices are only accessible through a network and the application design has to take this into account. Also, it is important to collect experience with networked device control, especially UPnP & DLNA control points, media servers and media renderers, before designing the application.

Response times of network-controlled devices can pose serious issues for an application. It is strongly recommended not to couple UI implementation and web API requests directly. A good application design should be prepared to handle response times of up to several seconds. The typical design for an application accessing the REST API will provide a multi-threaded request queue to e.g. load a media server directory and provide state machines for media renderer control, to prevent the application from blocking and to allow a user to keep interacting with the application.

Another important design consideration is the amount of metadata involved. Media servers can easily have directories with more than 10000 items. For most applications, especially on mobile and embedded devices, it's impossible to store the related metadata in memory (e.g. in a list box). In addition, the loading time for all the data would be unacceptable for a user trying to interact with the application. Therefore the application design should allow to only load the small number of visible items in a list view, plus possibly a cache for next few items to allow smooth scrolling. Loading this data should be done through the above mentioned multi-threaded request queue. The application should show some intermediate text or feedback to the user when scrolling, or jumping, too fast (e.g. each line shows „...") and then refresh the displayed text or graphic as soon as the request queue has processed and the respective media item has been loaded. Invisible items in the list view should release their (test) data to recover memory. This could be done by a background thread that's checking for text items out of the currently visible windows of items.

# REST API

The REST API consists of two modules that enable developers to utilize the underlying DLNA stack through standard HTTP requests. One module provides the functionality to browse servers and renderers and the according queues (available at http://..../nmc/rss). The data is returned either as RSS XML or as JSON. The second module enables the developer to utilize the APIs to select and control renderers and to manage the play queues. This module (available at http://..../nmc/rpc) exposes the APIs via HTTP GET requests and a specific syntax in the URLs.

Together with Twonky Server an example web-based JavaScript user interface implementation is provided (available at http://..../nmc/web), which should be used as a reference for the correct usage of the API. The web UI of Twonky Server itself contains a control point fully based on these interfaces and is hence the best example how to interact with them.

## Access Restrictions

Access to the REST API can be configured with the INI property **enablenmcwebapi**.

Possible values:

- 0: disabled
- 1: enabled (default)
- 2: local access only (from 8.1.2 on)

**Note:** Since 8.1.2 this property cannot be changed during run-time using an HTTP RPC call anymore. The INI file or command-line arguments are expected to contain the correct value at startup of Twonky Servers or client.

The property can be changed at any time via NMC IOCTL NMC_IOCTL_SET_INI_PROPERTY / "SetIniProperty" and takes effect immediately unless the initial value was 0. In this case the property can be changed, but has no effect as the according handlers are not available.

# RSS

## Overview

The RSS module provides an API to browse all discovered servers, renderers and their associated play queues in the network. This RSS based API can be utilized to develop non-UPnP client devices within the home network, as well as remote clients, to access media servers and all their shared content and renderers, which can be monitored and controlled.

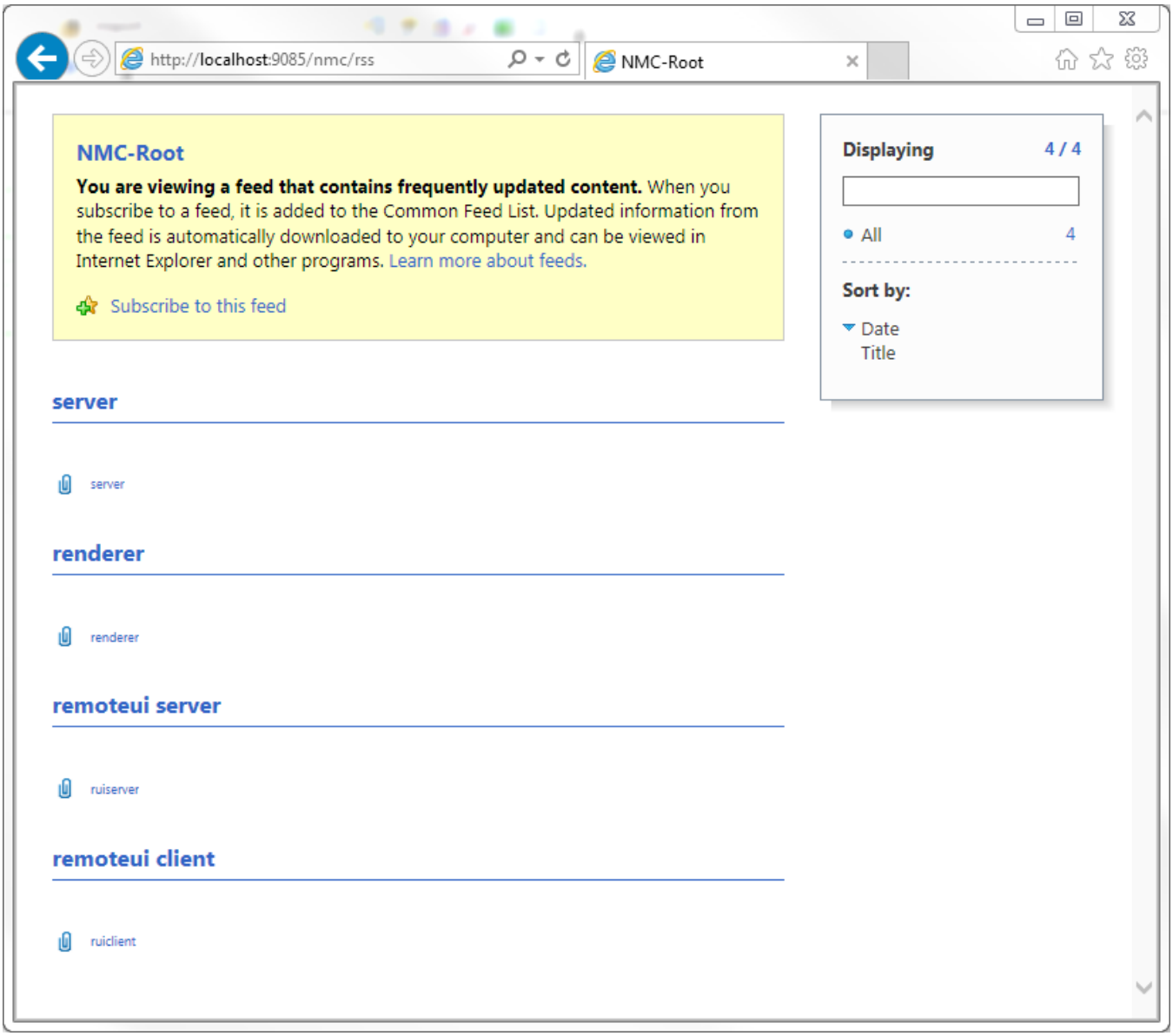
## Accessing the root level

The root level is the highest level RSS feed container.

To access the root level, the URL is:

```
Client SDK:         http://127.0.0.1:9085/nmc/rss
Embedded in server: http://127.0.0.1:9000/nmc/rss
```

The following screen shows the root level of the RSS feeds.




## Limiting Response Size

Since many of the responses to the API calls can be quite lengthy, there are two parameters that should be added to control the response size:

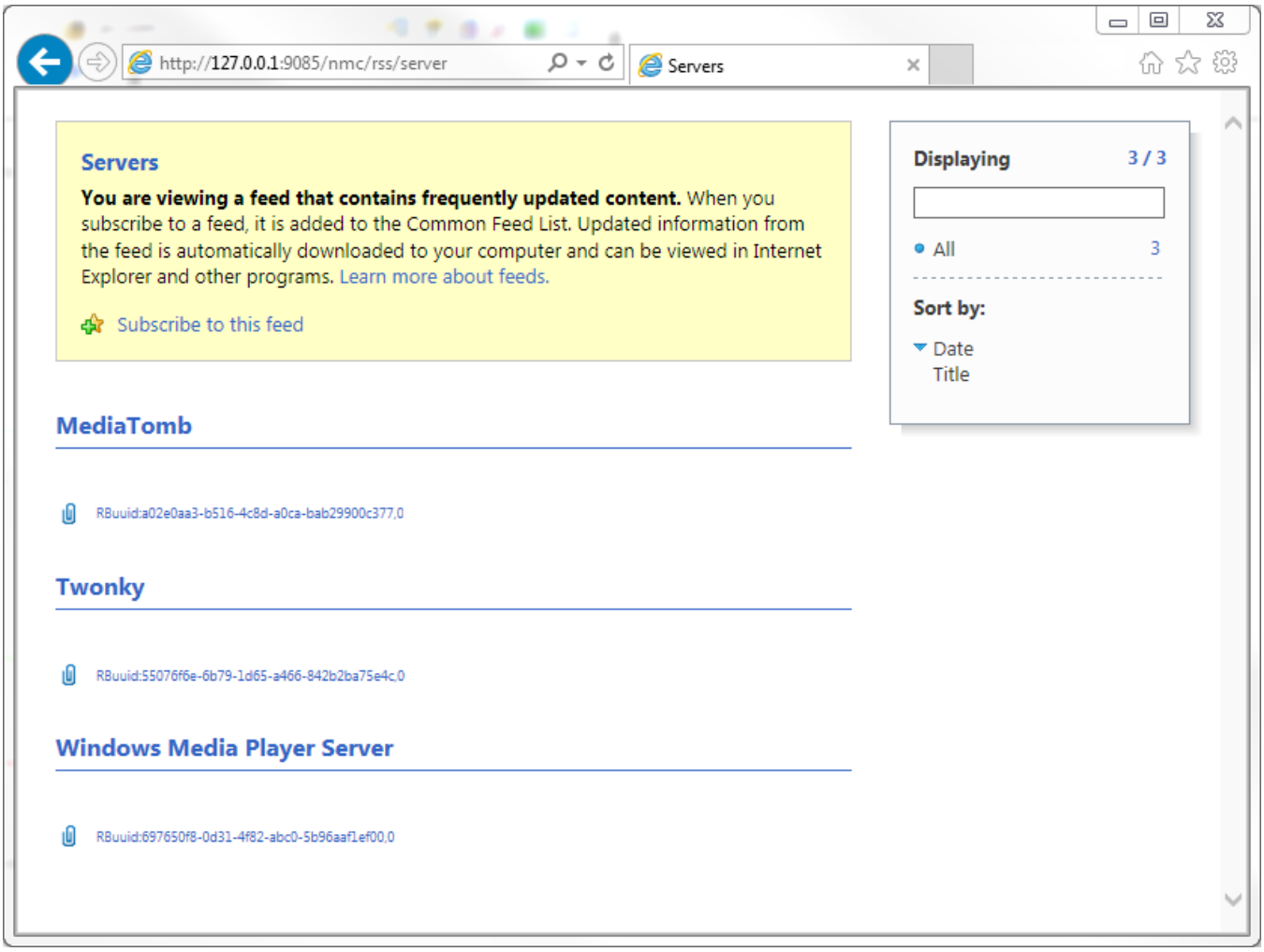
| URL Query Parameter | What it does |
|---|---|
| start= | Controls which item returned as the first item in the RSS feed. |
| count= | Controls how many items are returned. |

## Server List

The server list is accessible via the following URL:

```
Client SDK:         http://127.0.0.1:9085/nmc/rss/server
Embedded in server: http://127.0.0.1:9000/nmc/rss/server
```

The following screen shows the list of the servers discovered:




The following is the listing (pretty-printed for readability) for the page above, containing the expected output for one server:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/" xmlns:dlna="urn:schemas-dlna-org:metadata-1-0/" xmlns:
pv="http://www.pv.com/pvns/" >
    <channel>
        <title>Servers</title>
        <link>http://pv.com</link>
        <pubDate>Mon, 18 May 2015 08:42:25 GMT</pubDate>
        <description>3 objects available in container</description>
        <returneditems>3 objects returned from container</returneditems>
        <language>en-us</language>
        <copyright>PacketVideo</copyright>
        <id>Servers</id>
        <upnp:class>object.container</upnp:class>
        <url>http://127.0.0.1:9085/nmc/rss/server</url>
        <childCount>3</childCount>
        <item>
            <title>Twonky</title>
            <enclosure url="http://127.0.0.1:9085/nmc/rss/server/RBuuid%3A55076f6e-6b79-1d65-a466-
842b2ba75e4c,0" type="application/rss+xml"/>
            <bookmark>uuid%3A55076f6e-6b79-1d65-a466-842b2ba75e4c,0</bookmark>
            <isOnline>true</isOnline>
            <server>
                <name>Twonky</name>
                <friendlyName>Twonky</friendlyName>
                <manufacturer>PacketVideo</manufacturer>
                <modelName>TwonkyServer</modelName>
```

```
                <modelNumber>8.1.0</modelNumber>
                <modelDescription>TwonkyServer (Windows, T0)</modelDescription>
                <dlnaVersion>DMS-1.50</dlnaVersion>
                <upnpVersion>1.0</upnpVersion>
                <playlistSupport>true</playlistSupport>
                <isLocalDevice>true</isLocalDevice>
                <isInternalDevice>false</isInternalDevice>
                <UDN>uuid:55076f6e-6b79-1d65-a466-842b2ba75e4c</UDN>
                <baseURL>http://127.0.0.1:9000/dev0</baseURL>
                <multiUserSupport>false</multiUserSupport>
                <dtcpSupport>false</dtcpSupport>
                <dtcpPushSupport>false</dtcpPushSupport>
                <dtcpCopySupport>false</dtcpCopySupport>
                <dtcpMoveSupport>false</dtcpMoveSupport>
                <uploadSupportAV>true</uploadSupportAV>
                <uploadSupportImage>true</uploadSupportImage>
                <uploadSupportAudio>true</uploadSupportAudio>
                <knownServer>Twonky 8.1.0</knownServer>
                <wellKnownBookmark realContainerId="0$1">.,music</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$8">.,music/all</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$9">.,music/playlists</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$10">.,music/genre</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$11">.,music/artists</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$12">.,music/albums</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$13">.,music/folders</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$14">.,music/rating</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2">.,picture</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$20">.,picture/all</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$21">.,picture/playlists</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$22">.,picture/folders</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$23">.,picture/date</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$24">.,picture/albums</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$25">.,picture/keywords</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$26">.,picture/rating</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3">.,video</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$27">.,video/all</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$28">.,video/playlists</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$29">.,video/genre</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$31">.,video/actors</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$32">.,video/series</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$33">.,video/folders</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$34">.,video/rating</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$15">.,music/artistindex</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$16">.,music/artistalbum</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$17">.,music/genrealbum</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$18">.,music/genreartistalbum</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$35">.,video/albums</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$45">.,music/audiobooks</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0">.,root</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$4">.,music/dlna</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$5">.,picture/dlna</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$6">.,video/dlna</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$7">.,source/folders</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$19">.,music/composers</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$30">.,video/date</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$36">.,video/titleindex</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$37">.,playlists</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$38">.,video/classified</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$39">.,picture/geo</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$40">.,music/radio</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$41">.,video/live_tv</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$42">.,music/online</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$2$43">.,picture/online</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$3$44">.,video/online</wellKnownBookmark>
                <wellKnownBookmark realContainerId="0$1$46">.,music/audiobooksall</wellKnownBookmark>
            </server>
            <upnp:class>object.container</upnp:class>
        </item>
    </channel>
</rss>
```

Header tags:

| Tag | Description |
|---|---|
| <title> | Contains a description of the current list |
| <pubDate> | Current time and date |
| <description> | Total number of items available |
| <returneditems> | Number of items returned |
| <childCount> | Same as description but containing the number only |
| <id> | ▪ Server list: "Servers"<br>▪ Renderer list: "Renderers"<br>▪ RUI Server list: "RemoteUI Servers"<br>▪ RUI Client list: "RemoteUI Clients"<br>▪ Container: Object ID of container<br>▪ Item: Object ID of parent container |
| <upnp:class> | UPnP class of current of current container. For generic containers such as server list "object.container". |
| <url> | URL of the current view |

**Table 1: Description of header tags on an RSS feed**

Server description tags:

| Tag | Description |
|---|---|
| <item> | Specifies the sub category or the item itself. |
| <title> | Name of the device |
| <enclosure url=> | URL into the root directory of the server. The root bookmark (/RB) is the server ID. |
| <bookmark> | The bookmark pointing onto the server |
| <isOnline> | Since 7.2: true if device is online, false if offline.<br>If offline, then device info can be retrieved, but it cannot be browsed or searched. |
| <server> | The server container has the following tags. |
| <name> | Server name |
| <friendlyName> | Server name (UPnP style) |
| <manufacturer> | Name of the manufacturer. |
| <modelName> | Server model name. |
| <modelNumber> | Server version. |
| <modelDescription> | Server description. |
| <dlnaVersion> | DLNA version, typically DMS-1.50 or M-DMS-1.50 |
| <upnpVersion> | UPnP AV version, typically 1.0 |
| <playlistSupport> | true\|false<br>This is set to true, if server supports these UPnP actions:<br>CreateObject, CreateReference, DestroyObject, UpdateObject |
| <isLocalDevice> | true\|false<br>This is set to true, if device is local (means on same machine, not necessarily part of the process). |
| <isInternalDevice> | true\|false<br>This is set to true, if device is in the same process, i.e. this is the server you got this response from. |

| | |
|---|---|
| <UDN> | The unique identifier for the device. |
| <baseURL> | The server's presentation page URL. |
| <multiUserSupport> | Gets information whether the media server has multi-user support enabled or not.<br><br>For Twonky Server 7.3 or later. For all other servers it is signalled as not being enabled. |
| <dtcpSupport> | Does the server generally support DTCP, ie. can it share content streamed via DTCP-IP? |
| <dtcpPushSupport> | Does server support DTCP push? |
| <dtcpCopySupport> | Does server support DTCP copy? |
| <dtcpMoveSupport> | Does server support DTCP move? If so, which formats? |
| <uploadSupportAV> | Does server support video upload? |
| <uploadSupportImage> | Does server support image upload? |
| <uploadSupportAudio> | Does server support audio upload? |
| <knownServer> | The type of server that is recognized.<br><br>If this is provided, then a list of well-known bookmarks is available for this server (see wellknownBookmark). |
| <wellKnownBookmark> | A list of well-known bookmarks supported on this server.<br>These bookmarks can be used to browse specific containers with certain known IDs.<br><br>Please note that these IDs are not necessarily those returned in browse responses.<br>Initially the client tries to retrieve the real IDs if these are not fixed.<br>If known, then this tag has an attribute realContainerId listing the actual object ID. |
| <upnp:class> | UPnP object type, for servers always object.container. |

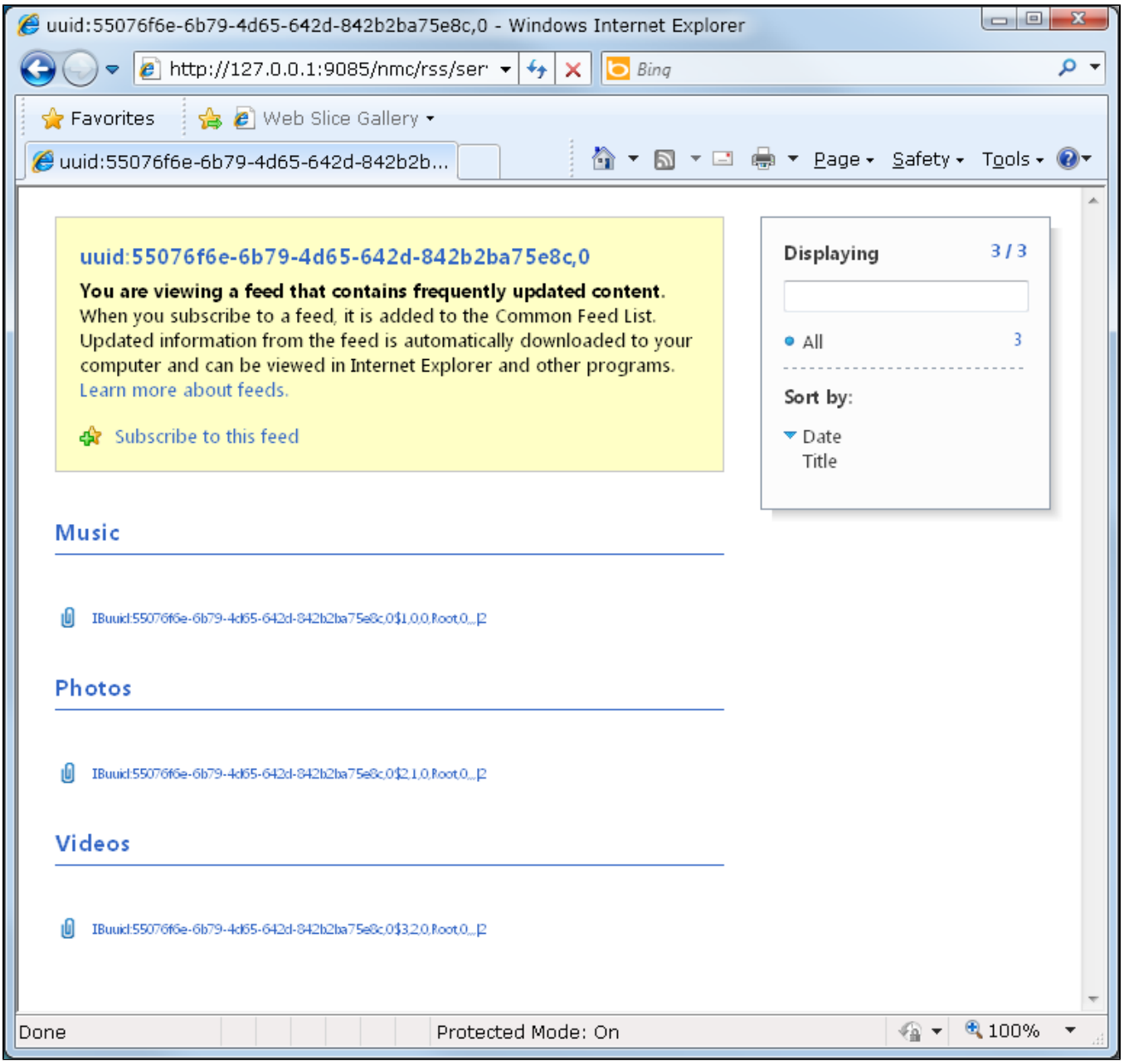
**Table 2: Description of server root item tags on an RSS feed container**

## Server Content

Every item contains the URL of the item in the enclosure tag. By using this it is possible to traverse the navigation tree of the server.

**Warning:** Always use paged browsing to list server content (see section *Limiting Response Size* above). Especially when browsing the *All* folders the number of items can easily exceed ten thousand items. Querying them all at once takes first a long time and second can lead to out of memory conditions. Hence, it is strongly recommended to use a paged browsing with some additional loading of additional pages in the background.

The next screen shows the root level of a Twonky Server.




The source for the server root level is:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/" xmlns:dlna="urn:schemas-dlna-org:metadata-1-0/" xmlns:
pv="http://www.pv.com/pvns/" >
<channel>
<title>BAIER980: baier:</title>
<link>http://pv.com</link>
<pubDate>Thu, 06 Oct 2011 11:30:27 GMT</pubDate>
<description>4 objects available in container</description>
<returneditems>4 objects returned from container</returneditems>
<language>en-us</language>
<copyright>PacketVideo</copyright>
<id>0</id>
<upnp:class>object.container</upnp:class>
<url>http://127.0.0.1:9085/nmc/rss/server/RBuuid:4f1bf61c-599b-443d-b41d-36408362ec75,0</url>
<item>
<title>Music</title>
```

<enclosure url="http://127.0.0.1:9085/nmc/rss/server/RBuuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,0/IBuuid%
3A4f1bf61c-599b-443d-b41d-36408362ec75,1,0,0,Root,0,,,%7C2" type="application/rss+xml"></enclosure>
<bookmark>uuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,1,0,0,Root,0,,,%7C2</bookmark>
<meta id="1" restricted="1" parentID="0" childCount="10" searchable="1"><dc:title>Music</dc:title><upnp:
class>object.container</upnp:class><upnp:writeStatus>NOT_WRITABLE</upnp:writeStatus><upnp:searchClass
includeDerived="1">object.item.audioItem</upnp:searchClass><upnp:searchClass includeDerived="0">object.
container.playlistContainer</upnp:searchClass><upnp:searchClass includeDerived="0">object.container</upnp:
searchClass><upnp:searchClass includeDerived="1">object.container.genre</upnp:searchClass><upnp:searchClass
includeDerived="0">object.container.storageFolder</upnp:searchClass><upnp:searchClass includeDerived="0"
>object.container.genre.musicGenre</upnp:searchClass><upnp:searchClass includeDerived="0">object.item.
audioItem.musicTrack</upnp:searchClass><upnp:searchClass includeDerived="0">object.container.album.
musicAlbum</upnp:searchClass><upnp:searchClass includeDerived="1">object.container.album</upnp:
searchClass><upnp:searchClass includeDerived="0">object.container.person.musicArtist</upnp:searchClass><
/meta>
<upnp:class>object.container</upnp:class>
</item>
<item>
<title>Videos</title>
<enclosure url="http://127.0.0.1:9085/nmc/rss/server/RBuuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,0/IBuuid%
3A4f1bf61c-599b-443d-b41d-36408362ec75,2,1,0,Root,0,,,%7C2" type="application/rss+xml"></enclosure>
<bookmark>uuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,2,1,0,Root,0,,,%7C2</bookmark>
<meta id="2" restricted="1" parentID="0" childCount="8" searchable="1"><dc:title>Videos</dc:title><upnp:
class>object.container</upnp:class><upnp:writeStatus>NOT_WRITABLE</upnp:writeStatus><upnp:searchClass
includeDerived="0">object.container.playlistContainer</upnp:searchClass><upnp:searchClass includeDerived="0"
>object.container.person.movieActor</upnp:searchClass><upnp:searchClass includeDerived="0">object.container<
/upnp:searchClass><upnp:searchClass includeDerived="1">object.item.videoItem</upnp:searchClass><upnp:
searchClass includeDerived="1">object.container.genre</upnp:searchClass><upnp:searchClass includeDerived="0"
>object.container.storageFolder</upnp:searchClass><upnp:searchClass includeDerived="0">object.item.videoItem.
videoBroadcast</upnp:searchClass><upnp:searchClass includeDerived="0">object.container.album.videoAlbum<
/upnp:searchClass><upnp:searchClass includeDerived="0">object.container.genre.movieGenre</upnp:
searchClass><upnp:searchClass includeDerived="1">object.container.album</upnp:searchClass></meta>
<upnp:class>object.container</upnp:class>
</item>
<item>
<title>Pictures</title>
<enclosure url="http://127.0.0.1:9085/nmc/rss/server/RBuuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,0/IBuuid%
3A4f1bf61c-599b-443d-b41d-36408362ec75,3,2,0,Root,0,,,%7C2" type="application/rss+xml"></enclosure>
<bookmark>uuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,3,2,0,Root,0,,,%7C2</bookmark>
<meta id="3" restricted="1" parentID="0" childCount="7" searchable="1"><dc:title>Pictures</dc:
title><upnp:class>object.container</upnp:class><upnp:writeStatus>NOT_WRITABLE</upnp:writeStatus><upnp:
searchClass includeDerived="0">object.container.playlistContainer</upnp:searchClass><upnp:searchClass
includeDerived="0">object.item.imageItem.photo</upnp:searchClass><upnp:searchClass includeDerived="0">object.
container.album.photoAlbum</upnp:searchClass><upnp:searchClass includeDerived="0">object.container</upnp:
searchClass><upnp:searchClass includeDerived="0">object.container.storageFolder</upnp:searchClass><upnp:
searchClass includeDerived="0">object.container.album.photoAlbum.dateTaken</upnp:searchClass><upnp:
searchClass includeDerived="1">object.container.album</upnp:searchClass><upnp:searchClass includeDerived="1"
>object.item.imageItem</upnp:searchClass></meta>
<upnp:class>object.container</upnp:class>
</item>
<item>
<title>Playlists</title>
<enclosure url="http://127.0.0.1:9085/nmc/rss/server/RBuuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,0/IBuuid%
3A4f1bf61c-599b-443d-b41d-36408362ec75,12,3,0,Root,0,,,%7C2" type="application/rss+xml"></enclosure>
<bookmark>uuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,12,3,0,Root,0,,,%7C2</bookmark>
<meta id="12" restricted="1" parentID="0" childCount="2" searchable="0"><dc:title>Playlists</dc:
title><upnp:class>object.container</upnp:class><upnp:writeStatus>NOT_WRITABLE</upnp:writeStatus></meta>
<upnp:class>object.container</upnp:class>
</item>
</channel>
</rss>

The item tags have the following meaning:

| Tag | Description |
| --- | --- |

| | |
|---|---|
| <item> | Specifies the sub category or the item itself. |
| <title> | Name of the sub container. |
| <enclosure url=> | URL of an item. The root bookmark is the server.The item bookmark is set to traverse through the navigation tree of the server. |
| <container><meta> | Containers generate RSS Feeds based on the object classes of their contents.<br>The <meta> tags can contain quite a variety of information returned by the server. |

**Table 3: Description of item tags on an RSS feed container**

The structure of the RSS feeds is described in more detail in section 7.2.1 of the TwonkyServer 6.0 "Technical Specification and APIs" and the usage of RSS feeds is described in section 7.2.2 of that document.

### Parent List

As you navigate into the Music, Photo, Video, or other subcontainers of the server, a list of parent containers and links will be provided at the end of the RSS output. The list proceeds from most-specific (deepest) to least specific (topmost) containers. This information will look like the following:

```
<parentList>
    <parent>
        <id>0$1$12</id>
        <title>Album</title>
        <url>http://127.0.0.1:9085/nmc/rss/server/RBuuid:55076f6e-6b79-4d65-642d-842b2ba75e8c,0/IBuuid:
55076f6e-6b79-4d65-642d-842b2ba75e8c,-,0,0,Root,0,,,0,0,Music,0$1,,,|3</url>
    </parent>
    <parent>
        <id>0$1</id>
        <title>Music</title>
        <url>http://127.0.0.1:9085/nmc/rss/server/RBuuid:55076f6e-6b79-4d65-642d-842b2ba75e8c,0/IBuuid:
55076f6e-6b79-4d65-642d-842b2ba75e8c,-,0,0,Root,0,,,|2</url>
    </parent>
    <parent>
        <id>0</id>
        <title>Root</title>
        <url>http://127.0.0.1:9085/nmc/rss/server/RBuuid:55076f6e-6b79-4d65-642d-842b2ba75e8c,0</url>
    </parent>
</parentList>
```

| Tag | Description |
|---|---|
| <id> | Container ID of the parent item |
| <title> | Name of the container |
| <url> | URL of the container |

### Sorting

By appending the **sort** or the **try_sort** parameter it can be requested that the server returns the items in a specific sort order.
The actual sorting happens on server side and it also depends on the server, which sort options are available.

The difference between both parameters is that **sort** will return an error if the server fails with that sort option whereas **try_sort** will reset to the server default sort-order and return that response instead.

The sort parameter takes a comma-separated list of sort options of this type:

```
sort_option=ascending|descending
```

Example:

```
http://127.0.0.1:9000/nmc/rss/server/RBuuid%3A55076f6e-6b79-4d65-6497-842b2ba75e4c,0%243/IB.,video/all?
sort=title=descending,creator=ascending
```

The example shows that the "All Videos" container shall be listed with title descending and and creator ascending.
Current known sort options are:

- title

- creator
- genre
- album
- artist

It is also possible to directly provide the UPnP names.
Example:

```
sort=-dc:title,+dc:creator
```

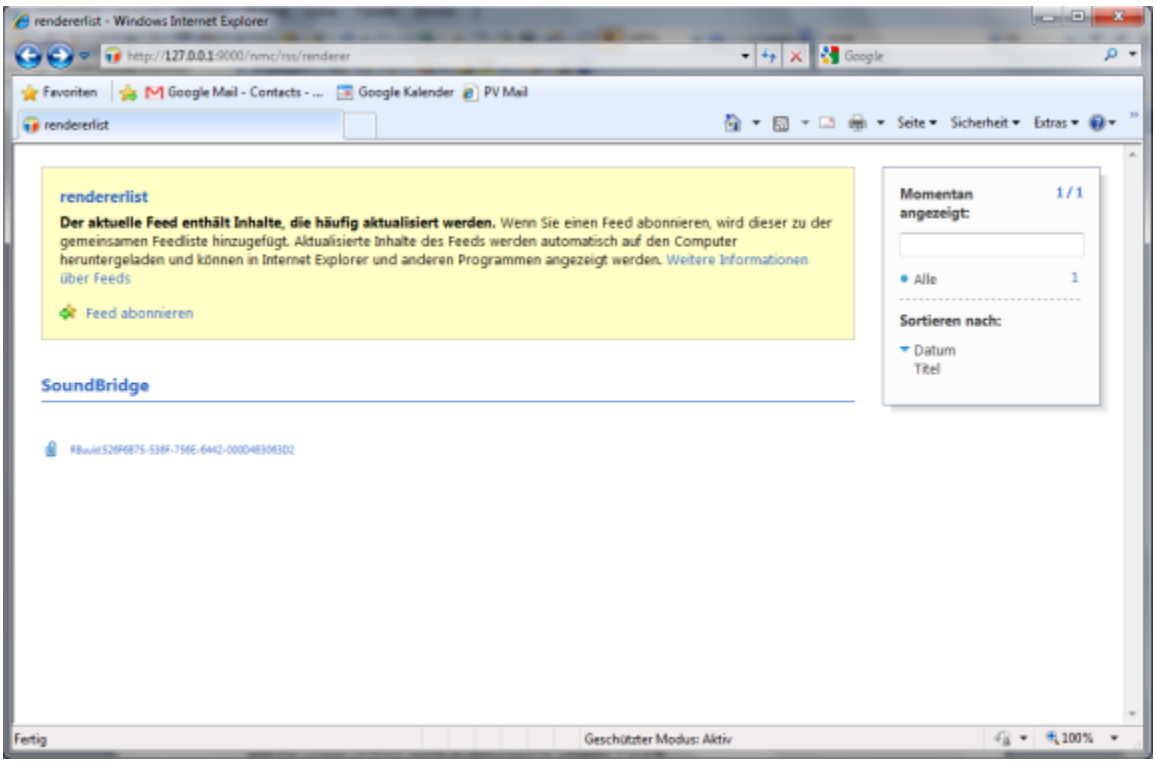
**Note:** Either use the first or the second option. When both schemes are provided, then all options using the UPnP naming scheme are ignored.

## Renderer List

The renderer list is accessible via the link on the root page. The URL is

```
http://127.0.0.1:9999/nmc/rss/renderer
```

.




The source for the renderer list is:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/" xmlns:dlna="urn:schemas-dlna-org:metadata-1-0/" xmlns:
pv="http://www.pv.com/pvns/" >
<channel>
<title>rendererlist</title>
<link>[http://pv.com]</link>
<pubDate>Fri, 16 Jul 2010 14:00:08 GMT</pubDate>
<description>1 items</description>
<returneditems>1 returned items</returneditems>
<language>en-us</language>
<copyright>PacketVideo</copyright>
<item>
<title>SoundBridge</title>
<enclosure url="http://127.0.0.1:9000/nmc/rss/renderer/RBuuid%3A526F6B75-536F-756E-6442-000D4B3063D2" type="
application/rss+xml"></enclosure>
<bookmark>uuid%3A526F6B75-536F-756E-6442-000D4B3063D2</bookmark>
<isOnline>true</isOnline>
<renderer>
<name>SoundBridge</name>
<friendlyName>SoundBridge</friendlyName>
<manufacturer>Roku</manufacturer>
<modelName>Roku SoundBridge M400</modelName>
<modelNumber>M400</modelNumber>
<modelDescription>Roku SoundBridge network music player</modelDescription>
<dlnaVersion>DMP-1.00</dlnaVersion>
<upnpVersion>1.0</upnpVersion>
<isLocalDevice>false</isLocalDevice>
<baseURL>http://192.168.1.34:80</baseURL>
<TrackURI>unknown track uri</TrackURI>
<TrackMetaData>
<item>
<upnp:class>object.item.audioItem.musicTrack</upnp:class>
</item>
</TrackMetaData>
</renderer>
</item>
</channel>
</rss>
```

The root renderer list constructs the header part that is color-coded as light blue and the item tags those are color coded as light green. Between items are set new line so it would be easier to follow the items.

Tags are:

| Tag | Description |
|---|---|
| <channel> | Channel (metadata) and its contents |
| <title> | Contains a description of the media Container. |
| <pubDate> | Date and time when response was being created |
| <description> | Tells how many sub categories are found. |
| <returneditems> | Same as above. |

**Table 4: Description of renderer header tag on an RSS feed.**

The item tags have the following meaning:

| Tag | Description |
|---|---|
| <item> | Specifies the sub category or the item itself. |
| <title> | Name of the sub container. |
| <enclosure url> | URL of a root item. The root bookmark (/RB) is the server ID. |

| | |
|---|---|
| <isOnline> | Since 7.2: true if device is online, false if offline.<br>If offline, then device info can be retrieved, but it cannot be controlled. |
| <renderer> | The renderer container has the following items. |
| <name> | Renderer name |
| <friendlyName> | Renderer name (UPnP style) |
| <manufacturer> | Name of the manufacturer. |
| <modelName> | Renderer model name. |
| <modelNumber> | Renderer version. |
| <modelDescription> | Renderer description. |
| <dlnaVersion> | DLNA version, typically DMR-1.50/-1.51 or M-DMR-1.50/-1.51 |
| <upnpVersion> | UPnP AV version, typically 1.0 |
| <isLocalDevice> | true\|false<br>This is set to true, if device is local (means on same machine, not necessarily part of the process).<br>See note below. |
| <isInternalDevice> | true\|false<br>This is set to true, if device is in the same process, ie. an LDMR.<br>See note below. |
| <baseURL> | Presentation page |
| <TrackURI> | URI of current track |
| <TrackMetaData> | Metadata of current track |

**Table 5: Description of renderer root item tags on an RSS feed container**

Note:

> For protocol adaptation LDMRs (embedded DMR) such as Apple TV and Roku the tags reveal an internal but not local device as these LDMRs actually control remote devices.

## Renderer Queue

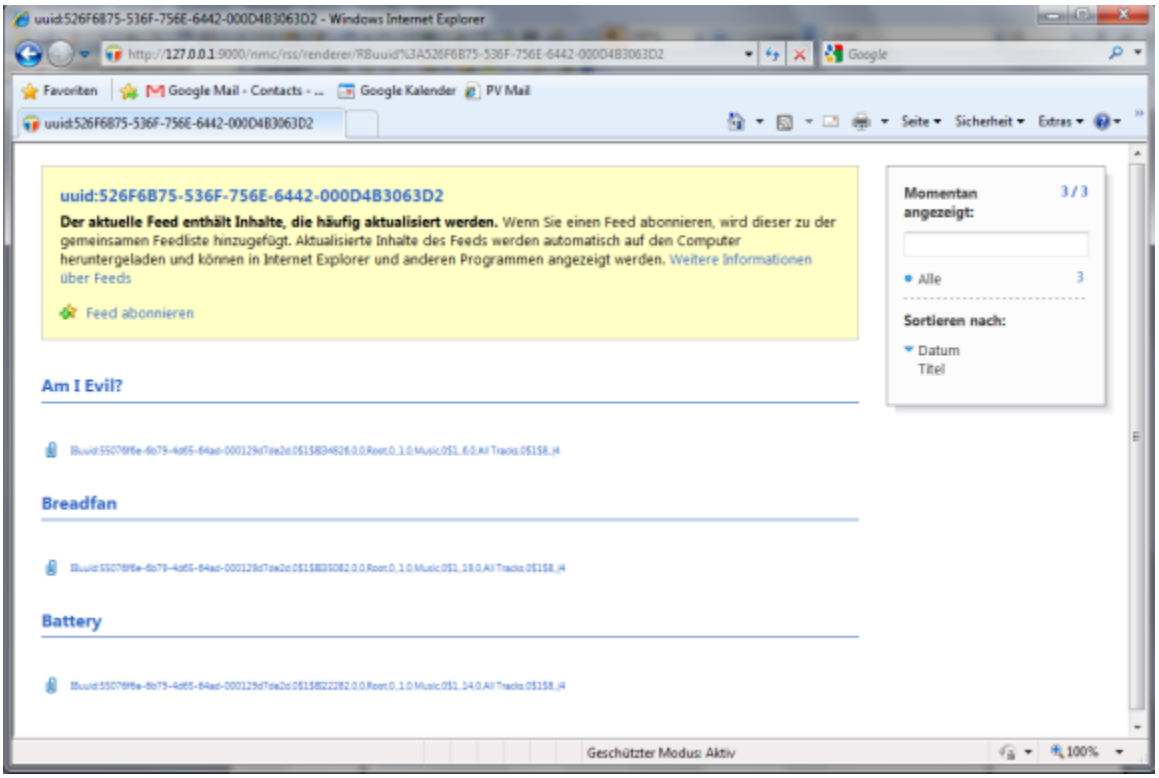
Every renderer item contains the URL of the renderer queue in the enclosure tag to retrieve the items listed in the renderer queue.

The renderer queue is located below the rendererlist by appending **/RB<renderer bookmark>**.

Example: http://127.0.0.1:9085/nmc/rss/RBuuid%3A56066f6e-6b79-1d65-a45b-842b2ba75e4c

**Warning:** Always use paged browsing to list a renderer queue (see section *Limiting Response Size* above). The number of items added by the user can easily exceed ten thousand items. Querying them all at once takes first a long time and second can lead to out of memory conditions. Hence, it is strongly recommended to use a paged browsing with some additional loading of additional pages in the background.

The next screen shows the queue of the Soundbridge.




The source for the renderer queue is:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/" xmlns:dlna="urn:schemas-dlna-org:metadata-1-0/" xmlns:
pv="http://www.pv.com/pvns/" >
<channel>
<title>uuid:526F6B75-536F-756E-6442-000D4B3063D2</title>
<link>[http://pv.com]</link>
<pubDate>Fri, 16 Jul 2010 13:50:30 GMT</pubDate>
<description>3 items</description>
<returneditems>3 returned items</returneditems>
<language>en-us</language>
<copyright>PacketVideo</copyright>
<item>
<title>Am I Evil?</title>
<enclosure url="http://127.0.0.1:9000/nmc/rss/renderer/RBuuid%3A526F6B75-536F-756E-6442-000D4B3063D2/IBuuid%
3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I34826,0,0,Root,0,,1,0,Music,0%241,,6,0,All%20Tracks,0%241%
248,,%7C4" type="application/rss+xml"></enclosure>
<bookmark>uuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I34826,0,0,Root,0,,1,0,Music,0%241,,6,0,All%
20Tracks,0%241%248,,%7C4</bookmark>
<meta restricted="1" parentID="0$1$8" id="0$1$8I34826">
<pv:supported>true</pv:supported>
<pv:duration>0:07:50</pv:duration>
<dc:title>Am I Evil?</dc:title>
<dc:date>1998-01-01</dc:date>
<upnp:genre>Metal</upnp:genre>
<upnp:album>Garage Inc.</upnp:album>
<upnp:originalTrackNumber>17</upnp:originalTrackNumber>
<dc:creator>Metallica</dc:creator>
<upnp:albumArtURI>[http://192.168.235.1:9000/disk/O0$1$8I34826.jpg?scale=160x160]</upnp:albumArtURI>
<upnp:artist>Metallica</upnp:artist>
<pv:extension>mp3</pv:extension>
<upnp:albumArtist>Metallica</upnp:albumArtist>
<pv:rating>1</pv:rating>
<pv:lastPlayedTime>2010-07-16T12:07:07</pv:lastPlayedTime>
<pv:playcount>8</pv:playcount>
<pv:modificationTime>1184772354</pv:modificationTime>
<pv:addedTime>1277751344</pv:addedTime>
```

<pv:lastUpdated>1184772354</pv:lastUpdated>
<res protocolInfo="http-get:*:audio/mpeg:DLNA.ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.
ORG_FLAGS=01703000000000000000000000000000" bitrate="128" size="7602176" duration="0:07:50">[http://192.
168.235.1:9000/disk/DLNA-PNMP3-OP01-FLAGS01703000/O0$1$8I34826.mp3]</res>
<upnp:class>object.item.audioItem.musicTrack</upnp:class>
</meta>
</item>
<item>
<title>Breadfan</title>
<enclosure url="http://127.0.0.1:9000/nmc/rss/renderer/RBuuid%3A526F6B75-536F-756E-6442-000D4B3063D2/IBuuid%
3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I35082,0,0,Root,0,,1,0,Music,0%241,,19,0,All%20Tracks,0%241%
248,,%7C4" type="application/rss+xml"></enclosure>
<bookmark>uuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I35082,0,0,Root,0,,1,0,Music,0%241,,19,0,All%
20Tracks,0%241%248,,%7C4</bookmark>
<meta restricted="1" parentID="0$1$8" id="0$1$8I35082">
<pv:supported>true</pv:supported>
<pv:duration>0:05:41</pv:duration>
<dc:title>Breadfan</dc:title>
<dc:date>1998-01-01</dc:date>
<upnp:genre>Metal</upnp:genre>
<upnp:album>Garage Inc.</upnp:album>
<upnp:originalTrackNumber>19</upnp:originalTrackNumber>
<dc:creator>Metallica</dc:creator>
<upnp:albumArtURI>[http://192.168.235.1:9000/disk/O0$1$8I35082.jpg?scale=160x160]</upnp:albumArtURI>
<upnp:artist>Metallica</upnp:artist>
<pv:extension>mp3</pv:extension>
<upnp:albumArtist>Metallica</upnp:albumArtist>
<pv:rating>2</pv:rating>
<pv:modificationTime>1184772354</pv:modificationTime>
<pv:addedTime>1277751344</pv:addedTime>
<pv:lastUpdated>1184772354</pv:lastUpdated>
<res protocolInfo="http-get:*:audio/mpeg:DLNA.ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.
ORG_FLAGS=01703000000000000000000000000000" bitrate="128" size="5531648" duration="0:05:41">[http://192.
168.235.1:9000/disk/DLNA-PNMP3-OP01-FLAGS01703000/O0$1$8I35082.mp3]</res>
<upnp:class>object.item.audioItem.musicTrack</upnp:class>
</meta>
</item>
<item>
<title>Battery</title>
<enclosure url="http://127.0.0.1:9000/nmc/rss/renderer/RBuuid%3A526F6B75-536F-756E-6442-000D4B3063D2/IBuuid%
3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I22282,0,0,Root,0,,1,0,Music,0%241,,14,0,All%20Tracks,0%241%
248,,%7C4" type="application/rss+xml"></enclosure>
<bookmark>uuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I22282,0,0,Root,0,,1,0,Music,0%241,,14,0,All%
20Tracks,0%241%248,,%7C4</bookmark>
<meta restricted="1" parentID="0$1$8" id="0$1$8I22282">
<pv:supported>true</pv:supported>
<pv:duration>0:05:10</pv:duration>
<dc:title>Battery</dc:title>
<dc:date>1993-01-01</dc:date>
<upnp:genre>Other</upnp:genre>
<upnp:album>Basel, Switzerland (06-20-1993</upnp:album>
<dc:creator>Metallica</dc:creator>
<upnp:artist>Metallica</upnp:artist>
<pv:extension>mp3</pv:extension>
<upnp:albumArtist>Metallica</upnp:albumArtist>
<pv:modificationTime>1098714096</pv:modificationTime>
<pv:addedTime>1277751343</pv:addedTime>
<pv:lastUpdated>1098714096</pv:lastUpdated>
<res protocolInfo="http-get:*:audio/mpeg:DLNA.ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.
ORG_FLAGS=01703000000000000000000000000000" bitrate="192" size="7438629" duration="0:05:10">[http://192.
168.235.1:9000/disk/DLNA-PNMP3-OP01-FLAGS01703000/O0$1$8I22282.mp3]</res>
<upnp:class>object.item.audioItem.musicTrack</upnp:class>
</meta>
</item>
</channel>
</rss>

Tags are:

| Tag | Description |
|---|---|
| <?xml version="1.0" encoding="utf-8"?> | Specifies the xml version and encoding |
| <rss version="2.0" | Version of RSS with the media parser with the xmlns Media Feature Tag. |
| <channel> | Channel (metadata) and its contents |
| <title> | Contains a description of the media Container. |
| <link> | A link to the RSS feed containing the media Container contents. |
| <pubDate> | Renderer startup time. |
| <description> | Tells how many sub categories are found. |
| <returneditems> | Same as above. |
| <language> | The language channel is written in. |
| <copyright> | Copyright notice for content in a channel. |

**Table 6: Description of renderer queue tag on an RSS feed.**

The item tags have the following meaning:

| Tag | Description |
|---|---|
| <item> | Specifies the sub category or the item itself. |
| <title> | Name of the sub container. |
| <enclosure url | URL of the item in the queue. The URL contains the renderer bookmark (/RB) and the item bookmark (/IB). |
| <bookmark> | The item bookmark. |
| <meta> | Containers generate RSS Feeds based on the object classes of their contents. |

**Table 7: Description of renderer queue item tags on an RSS feed container**

## Specifying Start, Count and Format

Parameters can be added at the end of each URL with a question mark "?" followed by the parameter keyword and the value. Further parameters are separated by an ampersand '&'. The parameters are:

```
start=[n]
count=[n]
fmt=json
[n] = number, min 0
```

Every call to such an RSS-URL returns a list of items. This list can be empty, has one or many items. The parameter 'start' defines the index of the first returned item and the parameter 'count' defines the number of items returned.
Example: The RSS URL (http://<RSS URL>) returns 45 items. The items should be placed on three pages, each page has place for 20 items. The parameters of each page are:

Page 1:

```
http://<RSS URL>?start=0&count=20 returns item0 to item19
```

Page 2:

```
http://<RSS URL>?start=20&count=20 returns item 20 to item 39
```

Page 3:

```
http://<RSS URL>?start=40&count=20 returns item 40 to 45
```

## JSON Formatted Data

The parameter 'fmt=json' returns the HTTP request in the format JSON (JavaScript Object Notation). JSON is a text format that is language independent.

### JSON Escaping

**NOTE:** Starting with version Twonky 8.1 are all JSON values by default **XML escaped**!

The XML escaping of the JSON feed returned by /nmc/rss?fmt=json and /nmc/rpc?fmt=json can be used if a web UI directly injects the returned values into HTML pages.

Example for disabled escaping:

```
{"title":"abc<script>alert('title')<\/script>def"}
```

Example for enabled escaping (default):

```
{"title":"abc&lt;script&gt;alert(&apos;title&apos;)&lt;\/script&gt;def"}
```

This setting affects all RPC and RSS results with JSON support. It does not affect any functions or non-JSON results.

The configuration can be configured in two ways:

1. Ini property escape_json / INI_ESCAPE_JSON
   This property can have values 0 (no escaping) and 1 (escaping). Default is 1.
   Note: This property is part of the blacklist which means it cannot be changed during run-time but only via the ini-file or command-line.
2. C-API function tm_nmc_rpc_set_json_escape()
   This toggles the setting during run-time and takes as parameter value either TRUE to enable escaping and FALSE to disable it.

As the escaping is security relevant (as can be seen from the example above), there is no RPC to change the setting.

### JSON Root Level

```
http://127.0.0.1:9085/nmc/rss?fmt=json
```


```
{
   "id":"NMC-Root",
   "title":"NMC-Root",
   "upnp:class":"object.container",
   "url":"http://127.0.0.1:9085/nmc/rss?fmt=json",
   "description":"2 objects available in container",
   "returneditems":"2 objects returned from container",
   "item":[
      {
         "title":"server",
         "enclosure":{
            "value":"",
            "type":"application/rss+xml",
            "url":"http://127.0.0.1:9085/nmc/rss/server"
         },
         "upnp:class":"object.container"
      },
      {
         "title":"renderer",
         "enclosure":{
            "value":"",
            "type":"application/rss+xml",
            "url":"http://127.0.0.1:9085/nmc/rss/renderer"
         },
         "upnp:class":"object.container"
      }
   ]
}
```

### JSON Renderer Queue

The URL of the renderer queue in the chapter above with the parameter 'fmt=json' is:

```
[http://127.0.0.1:9000/nmc/rss/renderer/RBuuid:526F6B75-536F-756E-6442-000D4B3063D2?fmt=json]
```

The source of the renderer queue in JSON-format is:


```
{"description":"3 items" ,
"returneditems":"3 returned items" ,
"item": [
{"title":"Am I Evil?" ,
"enclosure": { "value":"" ,"type":"application/rss+xml" ,"url":"http://127.0.0.1:9000/nmc/rss/renderer
/RBuuid%3A526F6B75-536F-756E-6442-000D4B3063D2/IBuuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I34826,
0,0,Root,0,,1,0,Music,0%241,,6,0,All%20Tracks,0%241%248,,%7C4" } ,
"bookmark":"uuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I34826,0,0,Root,0,,1,0,Music,0%241,,6,0,All%
20Tracks,0%241%248,,%7C4" ,
"meta": { "value":"" ,"id":"0$1$8I34826" ,"parentID":"0$1$8" ,
"restricted":"1" ,
"pv:supported":"true" ,
"pv:duration":"0:07:50" ,
"dc:title":"Am I Evil?" ,
"dc:date":"1998-01-01" ,
"upnp:genre":"Metal" ,
"upnp:album":"Garage Inc." ,
"upnp:originalTrackNumber":"17" ,
"dc:creator":"Metallica" ,
"upnp:albumArtURI":"http://192.168.17.1:9000/disk/O0$1$8I34826.jpg?scale=160x160" ,
"upnp:artist":"Metallica" ,
"pv:extension":"mp3" ,
"upnp:albumArtist":"Metallica" ,
"pv:rating":"1" ,
"pv:lastPlayedTime":"2010-07-16T12:07:07" ,
"pv:playcount":"8" ,
"pv:modificationTime":"1184772354" ,
"pv:addedTime":"1277751344" ,
"pv:lastUpdated":"1184772354" ,
"res": { "value":"http://192.168.17.1:9000/disk/DLNA-PNMP3-OP01-FLAGS01703000/O0$1$8I34826.mp3" ,
"duration":"0:07:50" ,
"size":"7602176" ,
"bitrate":"128" ,
"protocolInfo":"http-get:*:audio/mpeg:DLNA.ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.
ORG_FLAGS=01703000000000000000000000000000" } ,
"upnp:class":"object.item.audioItem.musicTrack"
}
}
,
{"title":"Breadfan" ,
"enclosure": { "value":"" ,"type":"application/rss+xml" ,"url":"http://127.0.0.1:9000/nmc/rss/renderer
/RBuuid%3A526F6B75-536F-756E-6442-000D4B3063D2/IBuuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I35082,
0,0,Root,0,,1,0,Music,0%241,,19,0,All%20Tracks,0%241%248,,%7C4" } ,
"bookmark":"uuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I35082,0,0,Root,0,,1,0,Music,0%241,,19,0,
All%20Tracks,0%241%248,,%7C4" ,
"meta": { "value":"" ,"id":"0$1$8I35082" ,"parentID":"0$1$8" ,
"restricted":"1" ,
"pv:supported":"true" ,
"pv:duration":"0:05:41" ,
"dc:title":"Breadfan" ,
"dc:date":"1998-01-01" ,
"upnp:genre":"Metal" ,
"upnp:album":"Garage Inc." ,
"upnp:originalTrackNumber":"19" ,
"dc:creator":"Metallica" ,
"upnp:albumArtURI":"http://192.168.17.1:9000/disk/O0$1$8I35082.jpg?scale=160x160" ,
"upnp:artist":"Metallica" ,
"pv:extension":"mp3" ,
"upnp:albumArtist":"Metallica" ,
"pv:rating":"2" ,
```

```
"pv:modificationTime":"1184772354" ,
"pv:addedTime":"1277751344" ,
"pv:lastUpdated":"1184772354" ,
"res": { "value":"http://192.168.17.1:9000/disk/DLNA-PNMP3-OP01-FLAGS01703000/O0$1$8I35082.mp3" ,
"duration":"0:05:41" ,
"size":"5531648" ,
"bitrate":"128" ,
"protocolInfo":"http-get:*:audio/mpeg:DLNA.ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.
ORG_FLAGS=01703000000000000000000000000000" } ,
"upnp:class":"object.item.audioItem.musicTrack"
}
}
,
{"title":"Battery" ,
"enclosure":
{"value":"" ,"type":"application/rss+xml" ,
"url":"http://127.0.0.1:9000/nmc/rss/renderer/RBuuid%3A526F6B75-536F-756E-6442-000D4B3063D2/IBuuid%
3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I22282,0,0,Root,0,,1,0,Music,0%241,,14,0,All%20Tracks,0%241%
248,,%7C4" \} ,
"bookmark":"uuid%3A55076f6e-6b79-4d65-64ad-000129d7de2d,0%241%248I22282,0,0,Root,0,,1,0,Music,0%241,,14,0,
All%20Tracks,0%241%248,,%7C4" ,
"meta": {"value":"" ,"id":"0$1$8I22282" ,"parentID":"0$1$8" ,
"restricted":"1" ,
"pv:supported":"true" ,
"pv:duration":"0:05:10" ,
"dc:title":"Battery" ,
"dc:date":"1993-01-01" ,
"upnp:genre":"Other" ,
"upnp:album":"Basel, Switzerland (06-20-1993" ,
"dc:creator":"Metallica" ,
"upnp:artist":"Metallica" ,
"pv:extension":"mp3" ,
"upnp:albumArtist":"Metallica" ,
"pv:modificationTime":"1098714096" ,
"pv:addedTime":"1277751343" ,
"pv:lastUpdated":"1098714096" ,
"res": \{ "value":"http://192.168.17.1:9000/disk/DLNA-PNMP3-OP01-FLAGS01703000/O0$1$8I22282.mp3" ,
"duration":"0:05:10" ,
"size":"7438629" ,
"bitrate":"192" ,
"protocolInfo":"http-get:*:audio/mpeg:DLNA.ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.
ORG_FLAGS=01703000000000000000000000000000" } ,
"upnp:class":"object.item.audioItem.musicTrack"
}
}
,
]
}
```


## Error Response

If an error happens during the RSS invocation, then a JSON object in this format is returned:

```
{"success": "false", "code": "<number>", "message": "<reason>"}
```

For example:

```
{"success": "false", "code": "3", "message": "Specified device does not exist"}
```

For a list of error codes see section *Response Messages and Codes* in the RPC chapter.

## Twonky server URL options

Parameter "download=1" can be added to URL of content shared by Twonky server.

Twonky will change content mimetype to "application/octet-stream" and add extra line to HTTP response header like this:

"Content-Disposition: attachment; filename=content_filename.ext"

This feature can be used to make browser show the "Save As" dialog box once a content link is clicked.

Example:

Requested URL:

http://192.168.0.1:9000/disk/DLNA-PNMP3-OP01-FLAGS01700000/O0$1$8I273.mp3?download=1

Response HTTP header

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 3515724
Date: Mon, 26 Jan 2015 09:14:10 GMT
Last-Modified: Wed, 07 Jan 2015 11:39:43 GMT
Accept-Ranges: bytes   Connection: close
Content-Disposition: attachment; filename="01-Watermark.mp3"
transferMode.dlna.org: Streaming
EXT:
Server: Windows NT/5.0, UPnP/1.0, pvConnect UPnP SDK/1.0,  Twonky UPnP SDK/1.1
```

## Requesting Metadata with Specific Adaptation

Twonky Server as well as Twonky SDK have support for client adaptations by the provided "Device Database".

The database resides in the directory "resources/devicedb" and contains adaptations for a large set of devices. The adaptations vary from dealing when beaming to specific renderers, over letting Twonky Server pretending being a different server to modyfying the metadata for certain clients.

For RSS the latter category is relevant as it may return the metadata with adaptations such as

- adapted and/or suppressed MIME types in resources
- specific flags to be added
- images, album art and thumbnails returned in certain resolutions
- etc.

By default the RSS server feed as well as RPC search results return all available metadata.

An application may override this **per request** by adding the **X-PV-CLIENTNAME** header entry. The syntax is:

*X-PV-CLIENTNAME: <display name of device db entry>*

If this is not provided, then the one in resources/devicedb/PacketVideo/Twonky_NMC_WebAPI.xml is used. If you look into the XML, then you find in it:

*<DisplayName>myTwonky</DisplayName>*

So specifying nothing is as if a request contains this header line:

*X-PV-CLIENTNAME: myTwonky*

You can use now the display name of any entry or add your own device db entry and use that name.

**WARNING**: In case you add your own device db entry:

- avoid a duplicate display name as this the key for the db, a second entry will be ignored
- if your adaptation is supposed to work also with Twonky Servers on other devices, please provide it back to PacketVideo to be incorporated in the regular release

This header entry is taken into account since version **8.1** and takes effect when browsing and searching Twonky Servers since version 7.2.

# RPC

## Call Syntax

The REST control API is used to control and manage servers and renderers, with the bulk of the calls being for the renderer.

The syntax of the call is:

```
http://127.0.0.1:9000/nmc/rpc/<function>?<parameter>
```

The syntax for two parameters is:

```
http://127.0.0.1:9000/nmc/rpc/<function>?<parameter1>&<parameter2>
```

In case the client is not part of the server, the default port is **9085**.

## Response Template

The general form of the RPC response will look something like the following.

```
{"success": "false", "code": "500", "message": "error on creating a context"}
```

The three JSON object properties are:

| Property | Meaning |
|---|---|
| success | Can be true or false. |
| code | Can be one of the standard HTTP Response Codes. |
| message | Can be a textual description of what occurred or the information that was requested. |

## Response Messages and Codes

General rules (with a few exceptions) for error codes:

- Negative error codes are generated by the RPC module
- Error codes zero and below hundred are generated by the Twonky stack.
- Error codes hundred and above are sent by the UPnP device.

| Code | Message |
|---|---|
| -11 | Download already in progress, cannot start another |
| -10 | Upload already in progress, cannot start another |
| -9 | Search failed |
| -8 | Internal error |
| -5 | Execution of the action failed |
| -4 | Bookmark not found |
| -3 | Error on deleting the context |
| -2 | Error on creating a context |
| -1 | Invalid args given for this NMC RPC call |
| 0 | Call succeeded |
| 1 | Target index does not exist |
| 2 | Parameter missing or invalid |
| 3 | Specified device does not exist |
| 4 | Known device that is currently not accessible |
| 5 | Specified device does not exist |
| 6 | Failed to connect to device |
| 7 | Invalid response from device |
| 8 | Out of memory |
| 9 | Query led to no result |
| 10 | Result does not fit into provided buffer |
| 11 | System is being shutdown, function aborted |
| 12 | Target already exists |
| 13 | Function / service not implemented |

| | |
|---|---|
| 14 | Local function failed for unknown reason |
| 20 | User aborted operation |
| 21 | Control Point rejected invocation, since this action is known to be broken |
| 22 | Resource could not be added due to limited space eg in an array |
| 23 | Connection was unexpectly closed or lost. |
| 24 | Could not get upload response code - success/failure of upload is unknown |
| 25 | Play queue is empty or contains only unsupported objects |
| 26 | Context Ambiguity found after network call |
| 27 | Timeout |
| 28 | Operation cancelled |
| 29 | Metadata not available |
| 30 | Browsing led to recursion |
| 50 | DTCP Move failed |
| 51 | DTCP Move receive failed |
| 52 | DTCP Move commit failed |
| 53 | Invalid DTCP Move resource |
| 54 | Not enough disk space to perform DTCP Move |
| 55 | Not supported content type for DTCP Move |
| 56 | Cannot create file system objects |
| 57 | Invalid DTCP Move target file name |
| 58 | Invalid DTCP Move session |
| 200 | HTTP request succeeded |
| 401 | Invalid SOAP Action |
| 402 | SOAP request with invalid arguments |
| 412 | State does not allow this type of request |
| 500 | Unknown internal device error |
| 501 | Action failed for unknown reason |
| 600 | Argument value invalid |
| 601 | Argument value out of range |
| 602 | Optional action not implemented |
| 603 | Out of memory |
| 701 | State does not allow requested action, Server does not find specified object, or Invalid Name. See UPNP RendernerControl Doc: 2.4.2.3. Errors |
| 702 | Update of object failed due to invalid supplied current value, or No content to be played |
| 703 | Update of object failed due to invalid new value, or Error reading media |
| 704 | Missing parameter, Unsupported playback format, or Recording Item doesn't exist |
| 705 | Property is read-only (DMS response), or Blocked by other CP or direct control (DMR response) |
| 706 | Parameters do not match |
| 708 | Unsupported or invalid search criteria |
| 709 | Unsupported or invalid sort criteria |
| 710 | Device does not support requested seek mode, or Server does not find specified container |

| | |
|---|---|
| 711 | Object is read-only (DMS response), or Cannot seek to requested track or position (DMR response) |
| 712 | Wrong metadata for object, or Play mode not supported |
| 713 | Parent directory of object is read-only |
| 714 | MIME type not supported |
| 715 | Resource access denied |
| 716 | The specified resource was not found |
| 717 | Play speed not supported |
| 718 | InstanceID not supported in AVTS (different for RCS) |
| 801 | No permission to access server/service |

## JSON Format

Where noted RPCs support as return value the JSON format by appending "fmt=json" to the URL.

See also JSON escaping for a discussion of the escaping of JSON values.

## Available RPCs

| Function | Parameter | Description |
|---|---|---|
| add_bookmark | renderer=[BOOKMARK]<br>item=[BOOKMARK]<br>index=[value] | Adds an item to the renderer queue at the position "index".<br>If no "index" parameter is provided, then the item will be added to the end of the queue. |
| add_metadata | renderer=[BOOKMARK]<br>metadata=[data]<br>url=[url]<br>index=[value]<br><br>alternatively to metadata the next fields can be used:<br><br>title=[value]<br>artist=[value]<br>album=[value]<br>genre=[value]<br>albumArtURI=[value]<br>duration= [value]<br><br>creator=[value]<br><br>addlHdrs=[value] | Adds an item to the renderer queue at the position "index".<br>If no "index" parameter is provided, then the item will be added to the end of the queue.<br><br>Please keep in mind that if this RPC is called asynchronously by multiple callers, the queueing order may be different than expected.<br><br>Instead of specifing metadata as a complete XML one can provide it as a set of fileds. In this case omit metadata parameter and make sure that you provided title parameter. Other metadata fields are optional.<br><br>If a local Twonky Server is running, then <*url*> may be a file URL, e.g.:<br><br>http://127.0.0.1:9085/nmc/rpc/add_metadata?renderer=uuid:07c8fc00-f5ff-43e0-9e67-0e9b330f8f0f&url=file:///share/videos/video.mp4<br><br>The RPC will contact the server for the corresponding bookmark and will then behave like RPC add_bookmark. Note that in this case any metadata parameters are ignored.<br><br><*addlHdrs*> (since 7.1.2):<br><br>Users of this API can pass "addlHdrs=<url_encoded_string>" which will then be passed to the Twonky ProxyServer to assist it in creating the metadata.<br>Additional headers to be added should be of the form:<br>addlHdrs=url.encode(X-TWONKY-PARAM: <param1>: <value1>\r\nX-TWONKY-PARAM: <param2>: <value2>\r\n)<br><br>Note that we do need the "\r\n" because these strings will be added as http headers as is. The "\r\n" would have to be encoded as "%0D%0A".<br><br>Starting with version 8.0 the parameter privateBM is ignored and the url parameter is also used as bookmark.<br><br>As of Twonky Server 8.1.1 Twonky ProxyServer is by default no longer part of the server and only available upon request. |
| beam | renderer=[index]<br>url=[url] | Beams the specified URL to the specified renderer. |

| | | |
|---|---|---|
| can_play | renderer=[BOOKMARK]<br>item=[BOOKMARK\|URL\|TYPE] | Queries whether the specified renderer supports the according item. The item parameter can have one of these types:<br><br>• a bookmark of an item to check if the according item is supported<br>• a complete protocol info such as "http-get:*:audio/mpeg:DLNA.ORG_PN=mp3"<br>• a MIME type such as "audio/mpeg"<br>• a URL pointing directly to a media item (this is neither resolved or followed)<br><br>Returns the string 1, if the renderer does not support the item, the string 0, if the renderer supports it.. |
| check_multiuser_access | server=<bookmark><br><br>user=<username><br><br>password=<password> | Checks whether the specified user has access to the server.<br><br>Returns a JSON object containing the result:<br><br>`{"success":"false"\|"true","code":"error code","message":"reason"}`<br><br>The following error codes are common:<br><br>• 0: Access is granted (user is known)<br>• 801: Access is denied (invalid credentials, server does not support multi-user)<br>• -1: Invalid arguments to RPC<br>• 1 and 3: Invalid device<br>• 7: Response from server was invalid<br>• 14: Could not contact server (no TLS, network issue)<br><br>The value of "success" is only true for error code 0, ie. when access is granted.<br><br>Note:<br><br>It is recommended that the multi user API is only used if the server metadata contains element "multiUserSupport" with value "true".<br><br>This is currently only supported for Twonky Server 7.3.<br><br>If the selected server is not in the same process, then the communication is done via https to hide the user credentials.<br><br>It is also recommended to use https when invoking this RPC.<br><br>Since 7.3 |
| clear | renderer=[BOOKMARK] | Clears the renderer queue. |
| delete_item | renderer=[BOOKMARK]<br>index=[value] | Deletes an item from the renderer queue. |
| download | bookmark=[BOOKMARK]<br>operation=[start \| stop \| getstatus]<br>filename=[path]<br><br>[fmt=json] | Downloads the specified bookmarked item. Operation parameter can start, stop, and get status of transfer. Filename parameter specifies the name of the file, however any path information provided here will be discarded and the file will ultimately be placed in the temporary directory used by the Client SDK or Twonky Server. This is "<appdata>/temp" (appdata as specified in the ini file).<br><br>If JSON output format requested, it will looks like this<br><br>{"result": "ok", "path":"<path to temp folder>/<filename>"} |

| | | |
|---|---|---|
| dtcp_content_upload<br>This functionality is only available for specific configurations of the Client SDK.<br>Please see „2.4 DTCP Support" for further details. | operation=[start \| stop \| getstatus]<br>src=[BOOKMARK]<br>dst=[BOOKMARK] | Controls a DTCP transfer between a source server and a destination server. Both servers must support DTCP. |
| dtcp_content_download<br><br>This functionality is only available for specific configurations of the Client SDK.<br>Please see „2.4 DTCP Support" for further details. | operation=[enabled \| start \| stop \| finish\| getstatus]<br>file=[BOOKMARK] | Controls DTCP-IP content download from the source server and the current server. Both servers must support DTCP. |
| dtcp_get_capabilities [1] | src=[BOOKMARK] | Returns the DTCP capability list of the specified server.<br>See tag description dlna:X_DLNACAP in chapter DTCP Support for the strings that can be returned. |
| get_albumart | renderer=[BOOKMARK]<br>or<br>renderer=[BOOKMARK]&index=[queue-index\|bookmark]<br>or<br>server=[BOOKMARK]<br><br>Optional parameters:<br>• max_edge=num_pixel default: 256<br>• mime_type=type<br><br>default: image/jpeg<br>• ignore_default=0\|1 default: 0<br>• scale=WxH<br>• device=0\|1 default: 0 | This function can be used to get the album art from these sources:<br><br>• from the metadata of the renderer of the currently playing item<br>• from the queue item specified by index or bookmark<br>• from the server using the specified item bookmark<br>• device icons for a server or renderer<br><br>The optional parameters can be used for these purposes:<br><br>• Specifying the maximum resolution of the album art. The call will only return album art with width and height below or equal to that size in pixel.<br>• Specifying the content type the album art shall have (typically image/jpeg or image/png).<br>• If ignore_default is set to 1, then album art delivered by Twonky server 5 or later that is detected as being the default album art will be ignored and an error will be returned. For all other servers is this setting ignored.<br>• Specifies a specific size for the image. If the server supports scaling, then the URL is accordingly modified. Format is "WxH". Both need to be equal to or smaller than max_edge. The returned image might be smaller to keep the aspect ratio of the thumbnail. If not supported by the server, then the non-scaled version is returned instead.<br>If not specified, then the unscaled version is returned.<br>• If device is set to 1 then the device icon for the specified renderer or server will be returned<br><br>The call returns the URL that fits best the parameters or an error, if these requirements cannot be met. The call works for album art of music and video items as well as thumbnails of image items. |

| | | |
|---|---|---|
| get_current_playspeeds | renderer=<BOOKMARK><br><br>Optional parameter:<br>• fmt=json | Returns the currently supported playspeeds from the renderer with the specified bookmark.<br><br>The according playspeeds are those announced in DLNA style in the CurrentTransportActions and are not taken from the AVTS service description.<br><br>In opposite to the DLNA specification contains this list always the value 1 to offer a complete list. If the renderer does not support or announce playspeed, then only the value 1 is returned.<br><br>In none-JSON format multiple values are separated by the pipe symbol.<br><br>Examples:<br><br>• RSS: -4\|-2\|-1\|-1/2\|1/2\|1\|2\|4<br>• JSON: {"PlaySpeeds": ["-4", "-2", "-1", "-1/2", "1/2", "1", "2", "4"]}<br><br>Since 7.2.6 |
| get_events | eventid=[last event ID] | Returns a list of server and renderer events separated by new line character "\r\n"<br><br>Result format: [event ID] "server"/"renderer" [event type] "bm="[BOOKMARK] "meta="[METADATA]<br><br>event ID: monotonically increasing integer signifying event ID<br><br>"server" / "renderer": string representing source of event<br><br>event type: integer signifying the type of event<br><br>"bm="[BOOKMARK]: bookmark of server/renderer<br><br>"meta=[METADATA]: any metadata. Currently used to provide container object id/bookmark in case of container update events if configured.<br><br>RPC returns current event ID as result if e.g. no event has yet recorded, state is same as requested event ID.<br><br>Note:<br><br>By default container update is disabled. To enable it IOCTL "SetContainerIDEvtMode" has to be invoked with one of these configuration parameters<br><br>"PASSTHROUGH" which returns comma separated list of updated container object IDs<br><br>"REPORT_EACH_ID" which returns each container object ID via separate container update events<br><br>"REPORT_BOOKMARK_ID" which returns each container bookmark via separate container update events<br><br>Updated Since 8.1 |
| get_item_path | server=[BOOKMARK]<br><br>Optional parameters:<br>• fmt=json | Gets from a local Twonky server the file path for an item referenced by the specified bookmark.<br><br>The function will return an error for remote servers or if the bookmark does not refer to a local item. The path of containers is not available.<br><br>The function returns a string containing an OS specific file path, not a file URL. If parameter "fmt=json" is provided, then the return format is the following:<br>{"path": ["<path>"]}<br><br>**Note:** Requires Twonky server 7.1 or later.<br><br>Example:<br>get_item_path?server=uuid:...&fmt=json<br>{"path": ["C:\Users\Public\Music\My Music.mp3"]} |
| get_known_bookmark_mapping | server=[BOOKMARK] | Returns a JSON object containing the mapping between well-known bookmarks such as ".,music" or ".,video/all", etc., and the container ID that represents them on the server. |
| get_mute | renderer=[BOOKMARK] | Returns the mute state.0 = off1 = on |
| get_nmc_version | | Returns the current version of the Client SDK / Twonky Server as string (e.g. 7.2) |

| | | |
|---|---|---|
| get_playindex | renderer=[BOOKMARK]<br><br>or<br><br>renderer=[BOOKMARK]&getIsPlayIndexValid=1 | Returns the string [playindex] \| [remaining]<br><br>Returns the string [playindex] \| [remaining]\|[OK/NA] |
| get_playmode | renderer=<BOOKMARK><br><br>Optional parameters:<br>• fmt=json | Retrieve the current play mode of the renderer.<br><br>See RPC play for a description of available modes.<br>Note: keep setup flag is not returned for this RPC as it makes no sense.<br><br>Return value:<br><br>Normal: Only the value is returned, eg. 0<br><br>JSON: {"mode": "<value>"}<br><br>The RPC is available since version 7.3.1. |
| get_playspeed | renderer=<BOOKMARK> | Returns the current playspeed of the specified renderer.<br><br>See also set_playspeed and play.<br><br>Since 7.2.6 |
| get_seek_capabilities | renderer=[BOOKMARK] | Returns on success a number:<br>• 0=no seek support<br>• 1=time seek support<br>• 2=byte seek support<br>• 3=time and byte seek support<br><br>In case of a failure a response like this is returned:<br><br>{"success": "false", "code": "2", "message": "Parameter missing or invalid"} |
| get_slideshow_delay | | Returns the global slideshow delay in seconds. |
| get_state | renderer=[BOOKMARK]<br><br>[&timeformat=numeric] | Returns the string [play state] \| [position] \| [duration]<br>Play state values are:<br>• 0=stopped<br>• 1=playing<br>• 2=preparing to play or seeking to new position<br>• 3=paused<br>• 6=no media (basically same as stopped)<br><br>Adding the timeformat=numeric parameter will change the position and duration format output from hh:mm:ss to a numeric format (in milliseconds). |
| get_status | | Indicates whether the the Web API is enabled. No response means no support. |
| get_supported_mimetypes | renderer=[BOOKMARK] [&fmt=json]<br><br>or server=[bookmark] [&fmt=json] [&sortcaps=1] or [&searchcaps=1] | Returns the protocol info of the renderer If json format is requested, then the return format is the following: {"renderer bookmark": ["info 1", "info2", ...]}<br><br>Returns server's search or sort capabilities.If json format is requested, then the return format is the following: {"server bookmark": ["info 1", "info2", ...]} |
| get_volume_db | renderer=[BOOKMARK] | Returns the volume in dB (Dezibel). |

| | | |
|---|---|---|
| get_volume_db_range | renderer=[BOOKMARK] | Returns the volume range in dB. |
| get_volume_percent | renderer=[BOOKMARK] | Returns the volume in percent (range 0 to 100). |
| log_clear | | Clear the logfile. |
| log_disable | mode=[0 \| 1] | Enable/disable logging. 0=enable 1=disable |
| log_getfile | | Returns the logfile contents. |
| log_set | sources=[value] level=[value] | Set the logging sources and level. The possible values see below. |
| move_to | renderer=[BOOKMARK] index=[value] toindex=[value] | Moves the item in the queue from the position "index" to the position "toindex". |
| pause | renderer=[BOOKMARK] resume=[0 \| 1] | Pause and restart playing. 0=pause 1=resume playing |
| play | renderer=<BOOKMARK><br><br>Optional parameters:<br><br>• mode=<mode><br>• playspeed=<playspeed> | Play the items in the renderer queue.<br><br>Each renderer has exactly one queue and that queue only controls that renderer (1-to-1). Starting playback on a renderer triggers the queue control as specified by the parameters. Starting the playback does not influence the playback on any other renderer. So multiple renderers can be queue controlled concurrently.<br><br>The parameter mode consists of play modes and flags. Flags can be combined with one play mode.<br><br>Play modes:<br><br>• 0: normal playback<br>• 1: shuffle (re-order) queue randomly before playback<br>• 2: random playback (order remains)<br><br>Flags:<br><br>• 0x100 (256): consume items after playback<br>• 0x200 (512): repeat queue<br>• 0x400 (1024): repeat single item<br>• 0x800 (2048): keep play mode setup as previously set via set_playmode (any other setting is ignored if this is set)<br><br>If the parameter is not provided, then 0 is used. The mode needs to be specified as decimal value and contain one play mode and several flags added together.<br><br>The playspeed parameter can be used to set the playspeed. Before using this the renderer should be queried about the currently available playspeeds using RPC get_current_playspeeds. The returned values of that RPC can be used as input.<br><br>If not provided, then playspeed "1" is used. See also RPC set_playspeed.<br><br>This parameter is available since 7.2.6.<br><br>See also set_playmode/get_playmode and set/get_playspeed.<br><br>See also play_from_position, set_playmode, skip_next, skip_previous, set_playindex |

| | | |
|---|---|---|
| play_from_position | renderer=<renderer-bookmark><br><br>Optional parameters:<br><br>• bookmark=<queue-item-bookmark><br>• startposms=<time-in-ms><br>• startposbyte=<byte-offset> | Starts normal playback of an item in the queue from a specific position.<br><br>Each renderer has exactly one queue and that queue only controls that renderer (1-to-1). Starting playback on a renderer triggers the queue control as specified by the parameters. Starting the playback does not influence the playback on any other renderer. So multiple renderers can be queue controlled concurrently.<br><br>Parameters:<br><br>• renderer: Bookmark of the target renderer to start playback on<br>• bookmark: Bookmark of the queue item to play. The item must be already in the queue. If multiple items with the same bookmark are in the queue, then the first matching one is taken. If not specified, then the first item in the queue is used.<br>• startposms: Start position in ms.<br>• startposbyte: Byte offset to start from.<br><br>The RPC tries several approaches to start playback from the specified time/byte position.<br><br>If none is supplied, then streaming starts from the beginning. If one of the positions is provided, then this is used. If both are provided, then time seek is used if supported, byte seek otherwise.<br><br>A zero or empty position is treated as if the parameter is not provided.<br><br>If the renderer supports only byte seek and byte position is not available, then a byte position is calculated from the time position. Similarly if it supports only time seek and time position is not available then time position is calculated from byte position.<br><br>Example: Start specific queue item at 120s<br><br>/nmc/rpc/play_from_position?renderer=uuid:56066f6e-6b79-1d65-a45b-842b2ba75e4c&bookmark=uuid:55076f6e-6b79-1d65-a466-842b2ba75e4c,_MCQxJDQ1JDc5MCQ3OTFSMjcw&startposms=120000&startposbyte=4000000<br><br>See also: reset_audiobook_position, play, set_playmode, skip_next, skip_previous, set_playindex<br><br>Since 8.1 |
| reset_audiobook_position | renderer=<renderer-bookmark><br><br>OR<br><br>server=<server-bookmark> | Since version 8.1 Twonky Server supports the concept of audiobooks. If such an audiobook is played to a renderer, then ca. every 30s the current playback position is stored on the source server to enable a later resume.<br><br>This RPC can be used to reset the audiobook playback position as if it was not played before. There is no distinction of users, ie. invoking the RPC resets the position for all users since as of now the position is not user specific. For servers not in the same process a network connection is required.<br><br>Note: Resetting the audiobook position does not lead to a server update event. But the RPC updates all caches. So re-retrieving the metadata of the audiobook container will no longer contain the play position.<br><br>The RPC removes from the audiobook container these properties:<br><br>• pv:playbackBookmark: Bookmark of audiobook track last played<br>• pv:playbackObjectId: Object ID of audiobook track last played<br>• pv:playbackTimeOffset: Last play position in ms for the refered audiotbook track<br>• pv:playbackByteOffset: Last play position in bytes for the refered audiobook track<br><br>Invocation with renderer bookmark:<br><br>• Resets the position of the audiobook currently in the renderer queue.<br>• This requires that the first item of the queue belongs to the audiobook to reset.<br>• This invocation can be used, if the target renderer that played the audiobook is still known and the queue has not been cleared.<br>• Example: /nmc/rpc/reset_audiobook_position?renderer=uuid:07c8fc00-f5ff-43e0-9e67-0e9b330f8f0f<br><br>Invocation with server bookmark:<br><br>• Bookmark to the audiobook container or a track of the audiobook to reset.<br>• This invocation can be used, if the UI stored the bookmark of the audiobook that was added to the queue.<br>• Example: /nmc/rpc/reset_audiobook_position?server=uuid:55076f6e-6b79-1d65-a466-842b2ba75e4c,_MCQxJDQ1JDgwNCQ4MDVSMjcw<br><br>If both parameters are provided, then the renderer parameter is ignored.<br><br>See: play_from_position<br><br>Since 8.1 |

| | | |
|---|---|---|
| search | server=[BOOKMARK]<br>search=[hex-encoded query]<br><br>Optional parameters:<br>• start=[value]<br>• count=[value]<br>• wkb=[well-known bookmark]<br>• sort=[sort criteria]<br>• fmt=json | Performs a search for the specified query. The search parameter is better described below.<br>Note: The search query must be hex-encoded.<br>For paging through a search, the start and count parameters can be used.<br><br>To search in a well-known bookmark, provide the server bookmark and the wkb parameter<br><br>For further description see chapter Search Syntax. |
| seek_bytes | renderer=[BOOKMARK]<br>seek=[value] | The seek position is relative to the beginning of the current stream in bytes.<br><br>Example: seek=50000 to seek to byte 50000 |
| seek_percent | renderer=[BOOKMARK]<br>seek=[value] | The seek position is relative to the beginning of the current stream in percent.<br><br>Example: seek=50 to seek to the middle of the stream |
| seek_time | renderer=[BOOKMARK]<br>seek=[value] | The seek position is relative to the beginning of the current stream in milliseconds.<br><br>Example: seek=50000 to seek to 50s |
| set_mute | renderer=[BOOKMARK]<br>mute=[0 \| 1] | Set mute on or off. 0=off 1 =on |
| set_playmode | renderer=<BOOKMARK><br><br>Optional parameters:<br>• mode=<mode> | Change the play mode on the fly.<br><br>The RPC can be used to either pre-configure the play mode or change it on the fly during playback.<br>To keep the play mode when invoking play RPC, provide value 2048 (keep setup) as play mode to the invocation.<br><br>If the mode parameter is missing, then mode 0 is used.<br><br>See RPC play for a description of available modes.<br>Note: keep setup flag is not available for this RPC as it makes no sense.<br><br>The RPC is available since version 7.3.1. |
| set_playspeed | renderer=<BOOKMARK><br><br>Optional parameter:<br>• playspeed=<playspeed> | Changes the playspeed of the current playback.<br><br>Before using this the renderer should be queried about the currently available playspeeds using RPC get_current_playspeeds. The returned values of that RPC can be used as input. If the parameter is not provided, then playspeed "1" is used.<br><br>See also get_playspeed and play.<br><br>Since 7.2.6. |
| set_brightness | renderer=[BOOKMARK]<br>brightness=[0..100] | Set brightness (since 7.0.10) |
| get_brightness | renderer=[BOOKMARK] | Get brightness (since 7.0.10) |

| | | |
|---|---|---|
| set_contrast | renderer=[BOOKMARK]<br><br>contrast=[0..100] | Set contrast (since 7.0.10) |
| get_contrast | renderer=[BOOKMARK] | Get contrast (since 7.0.10) |
| set_loudness | renderer=[BOOKMARK]<br><br>loudness=[0\|1] | Set loudness (since 7.0.10) |
| get_loudness | renderer=[BOOKMARK] | Get loudness (since 7.0.10) |
| set_playindex | renderer=[BOOKMARK]<br>index=[value] | Skips to the queue item at the specified index (index 0 is the first item in the queue).<br><br>Any current playback is interrupted and the item at the given index is continued. If not playing yet, then playback is started using the mode previously set via set_playmode or normal play mode if none was explicitly set.<br><br>Note: If the requested item is not supported, then playback stops.<br><br>See also: play, skip_next, skip_previous |
| skip_next | renderer=[BOOKMARK]<br><br>Optional parameters:<br>• index=[value]<br>• fmt=json | Skips to the next item in the queue.<br><br>The current playback is interrupted and the next supported item in the queue is played on the renderer.<br><br>In opposite to set_playindex allows this RPC a smoother transition to the next track. For example in random playback the invocation of this RPC will pick a random unplayed track and not the specified track.<br>Additionally this RPC uses the SetNextAVTransportURI/Next feature if supported by the according renderer. This usually leads to a faster transition to the next track and avoids the renderer going shortly back to the menu.<br><br>The *index* parameter is only used if the queue is not active, ie. it is not yet driven by this control point:<br><br>• If the queue is inactive and the index is given, then the RPC behaves as if set_playindex had been invoked.<br>• If the queue is active, then the index parameter is ignored.<br>• If the queue is inactive and the index parameter is not provided, then error 31 / "Queue is not active" will be returned.<br>• If the requested item is not supported, then playback stops.<br><br>Index 0 is the first item in the queue.<br><br>Behavior if queue is active:<br><br>• If the queue is active and skip fails then queue is stopped.<br>• If skip is done in paused state then state will remain paused<br>• Skips to the next supported queue item, not necessarily to the next item in the queue.<br>• If there are no further supported queue items in the queue, then this skips only to the beginning, if repeat mode was enabled.<br>• In shuffle mode is the queue reshuffled upon the next repeat cycle.<br>• In random play mode is the next item picked randomly.<br><br>Return value if successful:<br><br>• The RPC returns the play index after the skip.<br>• RSS format: Simply the index<br>• JSON format: {"playindex":"<value>"}<br><br>Example:<br><br>http://localhost:9085/nmc/rpc/skip_next?renderer=uuid:56066f6e-6b79-1d65-a45b-842b2ba75e4c&index=5&fmt=json<br><br>{"playindex":"5"}<br><br>Note: The index=5 paramater is given by the caller in case the queue is not active.<br><br>See: play, set_playmode, skip_previous, set_playindex<br><br>Since 8.1.1 |

| | | |
|---|---|---|
| skip_previous | renderer=[BOOKMARK]<br><br>Optional parameters:<br><br>• index=[value]<br>• fmt=json | Skips to the previous item in the queue.<br><br>The current playback is interrupted and the previous supported item in the queue is played on the renderer.<br><br>The *index* parameter is only used if the queue is not active, ie. it is not yet driven by this control point:<br><br>• If the queue is inactive and the index is given, then the RPC behaves as if set_playindex had been invoked.<br>• If the queue is active, then the index parameter is ignored.<br>• If the queue is inactive and the index parameter is not provided, then error 31 / "Queue is not active" will be returned.<br>• If the requested item is not supported, then playback stops.<br><br>Index 0 is the first item in the queue.<br><br>Behavior if queue is active:<br><br>• If the queue is active and skip fails then queue is stopped.<br>• If skip is done in paused state then state will remain paused.<br>• Skips to the next supported queue item before the current. If there are no further supported queue items in the queue, then this skips only to the end, if repeat mode was enabled.<br>• In shuffle mode is the queue reshuffled upon the next repeat cycle.<br>• For random playback there is a limited playback history of ten items. If there are items pending in this list, then the skip previous will skip to the previous object in the history. Otherwise it skips to the beginning of the current track. If the history is empty and the current track was deleted, then a random object is taken. The current object is tagged as not played and may be played again in the current round.<br><br>Return value if successful:<br><br>• The RPC returns the play index after the skip.<br>• RSS format: Simply the index<br>• JSON format: {"playindex":"<value>"}<br><br>Example:<br><br>http://localhost:9085/nmc/rpc/skip_previous?renderer=uuid:56066f6e-6b79-1d65-a45b-842b2ba75e4c&index=5&fmt=json<br><br>{"playindex":"5"}<br><br>Note: The index=5 paramater is given by the caller in case the queue is not active.<br><br>See: play, set_playmode, skip_next, set_playindex<br><br>Since 8.1.1 |
| ~~set_remote_access~~ | server=[BOOKMARK] enable=0\|1\|2 | Configures remote access for the myTwonky portal of a Twonky server 7.1.2 or later.<br><br>[BOOKMARK] is the bookmark or UDN of the server to configure.<br><br>enable contains the configuration for remote access of the server:<br><br>• 0: to disable remote access (default)<br>• 1: to make only this server available to the remote access tunnel<br>• 2: all servers within the network are tunneled through the specified server<br><br>Return error codes:<br><br>• -4 (invalid bookmark): Bookmark has an invalid format<br>• -1 (invalid args): Arguments are missing, enable has an invalid value, bookmark refers to a non-Twonky server<br>• 0 (ok): Value set successfully<br>• 3 (device gone): Specified server was not found<br>• 8 (out of memory): Out of memory<br>• 13 (not implemented): Remote access is not available on this server, most likely because it is an older version<br><br>Since 7.1.2<br><br>**Discontinued in 7.3** |

| | | |
|---|---|---|
| ~~get_remote_access~~ | Mandatory:<br>• server=[BOOKMARK]<br><br>Optional:<br>• fmt=json | Retrieves the remote access configuration for the myTwonky portal of a Twonky server 7.1.2 or later.<br><br>[BOOKMARK] is the bookmark or UDN of the server to configure.<br><br>If "fmt=json" is provided, then the return value is embedded into JSON.<br><br>Return value in case of success:<br>• 0: remote access disabled<br>• 1: only this server is available to the remote access tunnel<br>• 2: all servers within the network are tunneled through the specified server<br><br>Return error codes in case of an error:<br>• -4 (invalid bookmark): Bookmark has an invalid format<br>• -1 (invalid args): Arguments are missing, bookmark refers to a non-Twonky server<br>• 3 (device gone): Specified server was not found<br>• 8 (out of memory): Out of memory<br>• 13 (not implemented): Remote access is not available on this server, most likely because it is an older version<br><br>Since 7.1.2<br><br>**Discontinued in 7.3** |
| set_slideshow_delay | delay=[seconds] | Sets the global slideshow delay in seconds.<br>The setting affects all renderers.<br>The setting affects any current slideshows. |
| set_volume_db | renderer=[BOOKMARK]<br>volume=[value] | Set volume in dB (Decibel). |
| set_volume_percent | renderer=[BOOKMARK] volume=[value] | Set volume. The range of the value is between 0 and 100. |
| stop | renderer=[BOOKMARK] | Stop playing. |
| upload | server=[BOOKMARK]<br>file=[value]<br>operation=[start \| stop \| getstatus] | Uploads a file to the given filename. The name is used as title for the upload and the extension is being used to derive the according content-type.<br><br>Can start and stop an upload using the specific operation options. Using the getstatus operation will show the ostensible number of bytes received vs. the total number to be received.<br><br>Note: Currently the upload size is limited to 2 MB. |
| create_item_bookmark | server=[BOOKMARK]<br>objectid=[value] | Creates item bookmark from any server bookmark and object ID. |
| can_group | renderer=[BOOKMARK]<br><br>slave=[BOOKMARK] | Checks if renderer can be added as a slave renderer |
| add_slave | renderer=[BOOKMARK]<br><br>slave=[BOOKMARK] | Add slave renderer |

| | | |
|---|---|---|
| remove_slave | renderer=[BOOKMARK]<br><br>slave=[BOOKMARK] | Remove slaver renderer |
| get_slaves | renderer=[BOOKMARK] | Get list of slave renderers |
| is_slave | renderer=[BOOKMARK] | Check if renderer in slave mode |
| is_master | renderer=[BOOKMARK] | Check if renderer is master (has slave renderers) |
| get_master | renderer=[BOOKMARK] | Get master renderer |
| set_group_volume | renderer=[BOOKMARK]<br><br>volume=[N] | Set volume (in percent) on group of renderers |
| get_group_volume | renderer=[BOOKMARK] | Get volume (in percent) on group of renderers |
| set_group_mute | renderer=[BOOKMARK]<br><br>mute=[0/1] | Set mute on group of renderers |
| get_group_mute | renderer=[BOOKMARK] | Get mute on group of renderers |
| disconnect | renderer=[BOOKMARK] | Disconnect renderer from master |
| ioctl_nmc | <nmc ioctl>[%20<param>] | Executes a common IOCTL. Param consists of IOCTL to be executed and IOCTL's param if any, separated by %20.<br><br>The ioctl name is the value of the ioctl define and not the define itself. Names are case-sensitive. See the NMC_IOCTL defines in tm_nmc_common.h of the Doxygen description for available common ioctls.<br><br>In case of an error or if the ioctl does not return a string result the according JSON result is returned as listed in section *Response Template* above. In case of success and if a non-empty response string is returned by the ioctl, then this string is directly returned.<br><br>Example:<br><br>http://127.0.0.1:9085/nmc/rpc/ioctl_nmc?SetLogSource%20169<br><br>Since 7.0.6<br><br><br>As of 8.2.1 the following IOCTLs are **forbidden** via RPC:<br><br>• SetNMCAppName<br>• SetLogFilename<br>• SetNMCDataDirectory<br>• StopNMC<br>• StartNMC<br><br>As of 8.2.1 the IOCTLs SetIniProperty and GetIniProperty are **rerouted** onto the Twonky Server RPCs /rpc/set_option and /rpc/get_option when the NMC RPCs are invoked on Twonky Server or the Twonky SDK with enabled Twonky Server. In this case the responses will have the server format and not the NMC format. Hence, it is recommended to use the server RPCs to set and get INI properties instead if available. See the *Twonky Server API* documentation for more information about server RPCs. |

| | | |
|---|---|---|
| ioctl_dms | server=<BOOKMARK><br><br><server ioctl>[%20<param>] | Executes Server IOCTL. First param is server bookmark. Second param is IOCTL to be executed with it's param if any, separated by %20.<br><br>The ioctl name is the value of the ioctl define and not the define itself. Names are case-sensitive. See the DMSCP_IOCTL defines in tm_dms_cp.h of the Doxygen description for available server ioctls.<br><br>In case of an error or if the ioctl does not return a string result the according JSON result is returned as listed in section *Response Template* above. In case of success and if a non-empty response string is returned by the ioctl, then this string is directly returned.<br><br>Example:<br><br>http://127.0.0.1:9085/nmc/rpc/ioctl_dms?server=uuid:55076f6e-6b79-1d65-a466-842b2ba75e4c&GetSystemUpdateID<br><br>Since 7.0.6<br><br>See also *ioctl_nmc* for special handling of specific IOCTLs. |
| ioctl_dmr | renderer=<BOOKMARK><br><br><renderer ioctl>[%20<param>] | Executes Renderer IOCTL. First param is renderer bookmark. Second param is IOCTL to be executed with it's param if any, separated by %20.<br><br>The ioctl name is the value of the ioctl define and not the define itself. Names are case-sensitive. See the DMRCP_IOCTL defines in tm_dmr_cp.h of the Doxygen description for available renderer ioctls.<br><br>In case of an error or if the ioctl does not return a string result the according JSON result is returned as listed in section *Response Template* above. In case of success and if a non-empty response string is returned by the ioctl, then this string is directly returned.<br><br>Example:<br><br>http://127.0.0.1:9085/nmc/rpc/ioctl_dmr? renderer=uuid:56066f6e-6b79-1d65-a4a2-b827eb718faf&DMRSetSlideshowDelay%208<br><br>Since 7.0.6<br><br>See also *ioctl_nmc* for special handling of specific IOCTLs. |
| delete_server_object | server=[BOOKMARK] | Delete object from CDS |
| set_metadata | server=[BOOKMARK]<br><br>key=[metadata key]<br><br>value=[metadata value]<br><br>[index=[ix]] | Set item's metadata (since 7.0.10)<br><br>Example: set item's title to "new title"<br><br>http://localhost:9085/nmc/rpc/set_metadata?server=<items bookmark>&key=dc:title&value="new title"<br><br>Note: in TS changing metadata alowed only for uploaded content. |
| dump_renderer_info | renderer=[bookmark] | returns information about renderer and it's state |
| get_online_status | device=[bookmark] | Returns online status of device as true/false |
| persist_device | device=[BOOKMARK]&persist=[BOOLEAN]<br>OR<br><br>key=[GatewayMac::DeviceIndex]&<br><br>persist=[BOOLEAN] | Sets or Resets device to persist unless explicitly removed<br><br>Example: http://localhost:9085/nmc/rpc/persist_device?device=uuid:xxx&persist=1<br><br>http://localhost:9085/nmc/rpc/persist_device?key=XX:XX:XX:XX:XX:XX::n&persist=1 |

| | | |
|---|---|---|
| get_persistent_device_list | | Returns persistent db list in xml format.<br><br>Example: http://127.0.0.1:9085/nmc/rpc/get_persistent_db_list |
| reset_specific | gateway=[GatewayMac] | Clears persistent db for specific gateway.<br>GatewayMAC is mac address of the gateway in XX:XX:XX:XX:XX:XX format.<br><br>Example: http://127.0.0.1:9085/nmc/rpc/reset_specific?gateway=XX:XX:XX:XX:XX:XX |

**Table 8: Description of RPC functions**

## Controlling HTTP Response Codes

```
useRealHttpCode=true
```

During development of the RPC handler, it was discovered that sending HTTP Response Codes other than "200 OK" was problematic when using JSONP to handle the returning data. By default, the RPC calls will return "200 OK" as the HTTP Response Code.

Setting the useRealHttpCode to true causes the true HTTP Response Code to be returned, so this value could be something other than "200 OK".

## Examples

To illustrate the RPC functions here are some examples:

### Retrieving A Renderer Bookmark

In order to control a renderer, the API user needs to retrieve a bookmark for it first. The bookmark serves as a unique identifier for the renderer to be controlled. To do this information, an RSS feed needs to be retrieved. RSS feeds and format are fully described in the RSS documentation.

Here is a sample entry from the renderer RSS feed:

```
<item>
    <title>NMC Local Sample DMR</title>
    <enclosure url="http://127.0.0.1:9085/nmc/rss/renderer/RBuuid%3A56076f6e-6b79-4d65-648c-842b2ba76114"
type="application/rss+xml"></enclosure>
    <bookmark>uuid%3A56076f6e-6b79-4d65-648c-842b2ba76114</bookmark>
    <renderer>
        <name>NMC Local Sample DMR</name>
        <friendlyName>NMC Local Sample DMR</friendlyName>
        <manufacturer>PacketVideo</manufacturer>
        <manufacturerURL>http://www.pv.com/</manufacturerURL>
        <modelName>TwonkyRenderer</modelName>
        <modelNumber>6.0.2</modelNumber>
        <modelDescription>TwonkyRenderer</modelDescription>
        <serialNumber>6.0.2</serialNumber>
        <isLocalDevice>true</isLocalDevice>
        <isInternalDevice>true</isInternalDevice>
        <TrackURI></TrackURI>
        <TrackMetaData></TrackMetaData>
    </renderer>
    <upnp:class>object.container</upnp:class>
</item>
```

The renderer bookmark, which needs to be passed to all of the renderer RPCs, is the string contained in the <bookmark> tag.

### Retrieving An Item Bookmark

Using the RSS feeds, the API will return item information using the following form:

```
<item>
    <title>Track 1</title>
    <enclosure url="http://127.0.0.1:9000/nmc/rss/server/RBuuid%3A55076f6e-6b79-4d65-6436-842b2ba76114,0
/IBuuid%3A55076f6e-6b79-4d65-6436-842b2ba76114,0%241%248I31755,0,0,Root,0,,,1,0,Music,0%241,,,265,0,All%
20Tracks,0%241%248,,,%7C4" type="application/rss+xml"></enclosure>
    <bookmark>uuid%3A55076f6e-6b79-4d65-6436-842b2ba76114,0%241%248I31755,0,0,Root,0,,,1,0,Music,0%241,,,
265,0,All%20Tracks,0%241%248,,,%7C4</bookmark>
    <meta id="0$1$8I31755" parentID="0$1$8" restricted="1">
        <pv:duration>0:02:27</pv:duration>
        <dc:title>Track 1</dc:title>
        <upnp:genre>Sound Track</upnp:genre>
        <upnp:album>Walk the line</upnp:album>
        <upnp:originalTrackNumber>1</upnp:originalTrackNumber>
        <dc:creator>Joaquin Phoenix</dc:creator>
        <upnp:albumArtURI dlna:profileID="JPEG_TN">http://127.0.0.1:9000/disk/DLNA-PNJPEG_TN-CI1-
FLAGS00f00000/defaultalbumart/nocover_audio.jpg/O0$1$8I31755.jpg?scale=160x160</upnp:albumArtURI>
        <upnp:artist>Joaquin Phoenix</upnp:artist>
        <pv:extension>mp3</pv:extension>
        <upnp:albumArtist>Joaquin Phoenix</upnp:albumArtist>
        <pv:modificationTime>1143984134</pv:modificationTime>
        <pv:addedTime>1318604825</pv:addedTime>
        <pv:lastUpdated>1143984134</pv:lastUpdated>
        <res duration="0:02:27" size="2349056" bitrate="128" protocolInfo="http-get:*:audio/mpeg:DLNA.
ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.ORG_FLAGS=01700000000000000000000000000000">http://127.0.0.1:9000/disk/DLNA-
PNMP3-OP01-FLAGS01700000/O0$1$8I31755.mp3</res>
        <upnp:class>object.item.audioItem.musicTrack</upnp:class>
        <pv:noAgg>1</pv:noAgg>
    </meta>
</item>
```

### Checking If A Renderer Can Play An Item (can_play)

A song shall be added to the queue of a Roku SoundBridge.
The bookmark to the song is verified with the web API to check if the SoundBridge supports the media format.

Syntax:

```
can_play?renderer=[BOOKMARK]&item=[BOOKMARK]
```

URL:

```
/nmc/rpc/can_play?renderer=uuid%3A526F6B75-536F-756E-6442-000D4B3063D2&item=uuid:55076f6e-6b79-4d65-646d-
000129d7de2d,0$1$12$172R31498,0,0,Root,0,,0,0,Music,0$1,,9,0,Album,0$1$12,,3,0,Garage Inc.,0$1$12$172,,\|5
```

Function returns:

```
"0"
```

The SoundBridge can play the song.

### Adding A Bookmark To A Renderer Queue (add_bookmark)

The song is added to the queue of the SoundBridge.

Syntax:

```
add_bookmark?renderer=[BOOKMARK]&item=[BOOKMARK]&index=[value]
```

URL:

```
/nmc/rpc/add_bookmark?renderer=uuid%3A526F6B75-536F-756E-6442-000D4B3063D2&item=uuid:55076f6e-6b79-4d65-646d-
000129d7de2d,0$1$12$172R31498,0,0,Root,0,,0,0,Music,0$1,,9,0,Album,0$1$12,,3,0,Garage Inc.,0$1$12$172,,
\|5&index=0
```

Function returns:

```
"ok"
```

The song was added successfully.

### Adding An Item Using A URL To A Renderer Queue (add_metadata)

A different song is added to the queue of the SoundBridge.

The metadata= parameter of the add_metadata method uses URL-encoded XML information in one of the following forms to add an item to the queue:

### A Full DIDL-Lite XML response from a BrowseMetadata SOAP call

```
<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
xmlns:dlna="urn:schemas-dlna-org:metadata-1-0/" xmlns:arib="urn:schemas-arib-or-jp:elements-1-0/" xmlns:
dtcp="urn:schemas-dtcp-com:metadata-1-0/" xmlns:pv="http://www.pv.com/pvns/" xmlns="urn:schemas-upnp-org:
metadata-1-0/DIDL-Lite/">
        <item id="O0$1$8I6666" parentID="O0$1$8" restricted="1">
                <dc:title>MP3-L-01</dc:title>
                <dc:date>2010-01-01</dc:date>
                <upnp:genre>Other</upnp:genre>
                <upnp:album>AP Collection Vol 4</upnp:album>
                <dc:creator>GeoSuPhat</dc:creator>
                <upnp:albumArtURI dlna:profileID="JPEG_TN">http://172.16.149.27:9000/disk/defaultalbumart
/nocover_audio.jpg/OO0$1$8I6666.jpg?scale=160x160</upnp:albumArtURI>
                <upnp:artist>GeoSuPhat</upnp:artist>
                <pv:extension>mp3</pv:extension>
                <upnp:albumArtist>GeoSuPhat</upnp:albumArtist>
                <pv:modificationTime>1285172443</pv:modificationTime>
                <pv:addedTime>1318801106</pv:addedTime>
                <pv:lastUpdated>1285172443</pv:lastUpdated>
                <res duration="1:54:00" size="273613952" bitrate="320" protocolInfo="http-get:*:audio/mpeg:
DLNA.ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.ORG_FLAGS=01700000000000000000000000000000">http://172.16.149.27:9000
/disk/DLNA-PNMP3-OP01-FLAGS01700000/OO0$1$8I6666.mp3</res>
                <upnp:class>object.item.audioItem.musicTrack</upnp:class>
        </item>
</DIDL-Lite>
```

### Part of a Full DIDL-Lite XML response

```
<item id="O0$1$8I6666" parentID="O0$1$8" restricted="1">
        <title>MP3-L-01</title>
        <date>2010-01-01</date>
        <genre>Other</genre>
        <album>AP Collection Vol 4</album>
        <creator>GeoSuPhat</creator>
        <albumArtURI profileID="JPEG_TN">http://172.16.149.27:9000/disk/defaultalbumart/nocover_audio.jpg
/OO0$1$8I6666.jpg?scale=160x160</albumArtURI>
        <artist>GeoSuPhat</artist>
        <extension>mp3</extension>
        <albumArtist>GeoSuPhat</albumArtist>
        <modificationTime>1285172443</modificationTime>
        <addedTime>1318801106</addedTime>
        <lastUpdated>1285172443</lastUpdated>
        <res duration="1:54:00" size="273613952" bitrate="320" protocolInfo="http-get:*:audio/mpeg:DLNA.
ORG_PN=MP3;DLNA.ORG_OP=01;DLNA.ORG_FLAGS=01700000000000000000000000000000">http://172.16.149.27:9000/disk
/DLNA-PNMP3-OP01-FLAGS01700000/OO0$1$8I6666.mp3</res>
        <class>object.item.audioItem.musicTrack</class>
</item>
```

### An RSS <meta> XML element (Music)

```
<meta id="0$1$12$69R266" refID="0$1$8I266" parentID="0$1$12$69" restricted="1">
        <duration>0:01:00</duration>
        <title>B-LPCM-1</title>
        <genre>Unknown</genre>
        <album>Unknown</album>
        <creator>Unknown</creator>
        <albumArtURI profileID="JPEG_TN">http://172.16.149.27:9000/disk/defaultalbumart/nocover_audio.jpg
/O0$1$8I266.jpg?scale=160x160</albumArtURI>
        <artist>Unknown</artist>
        <extension>pcm</extension>
        <modificationTime>1237455478</modificationTime>
        <addedTime>1318801106</addedTime>
        <lastUpdated>1237455478</lastUpdated>
        <res duration="0:01:00" size="5303806" protocolInfo="http-get:*:audio/L16;rate=44100;channels=1:DLNA.
ORG_PN=LPCM;DLNA.ORG_OP=01;DLNA.ORG_FLAGS=01700000000000000000000000000000">http://172.16.149.27:9000/disk
/DLNA-PNLPCM-OP01-FLAGS01700000-rate44100-channels1/O0$1$8I266.pcm</res>
        <class>object.item.audioItem.musicTrack</class>
</meta>
```

### An RSS <meta> XML element (Photo)

```
<meta id="0$2$24$2214R523" refID="0$2$20I523" parentID="0$2$24$2214" restricted="1">
        <title>P1050070</title>
        <date>2008-11-18</date>
        <album>Hawaii</album>
        <extension>JPG</extension>
        <rating>4</rating>
        <modificationTime>1321621829</modificationTime>
        <addedTime>1322752201</addedTime>
        <lastUpdated>1321621829</lastUpdated>
        <orientation>1</orientation>
        <res size="3083986" resolution="2448x3264" protocolInfo="http-get:*:image/jpeg:DLNA.ORG_PN=JPEG_LRG;
DLNA.ORG_OP=01;DLNA.ORG_FLAGS=00f00000000000000000000000000000">http://172.16.149.42:9000/disk/DLNA-
PNJPEG_LRG-OP01-FLAGS00f00000/O0$2$20I523.JPG</res>
        <res size="3812" resolution="120x160" protocolInfo="http-get:*:image/jpeg:DLNA.ORG_PN=JPEG_TN;DLNA.
ORG_OP=01;DLNA.ORG_FLAGS=00f00000000000000000000000000000">http://172.16.149.42:9000/disk/DLNA-PNJPEG_TN-
OP01-FLAGS00f00000/O0$2$20I523.JPG?scale=120x160</res>
        <res resolution="360x480" protocolInfo="http-get:*:image/jpeg:DLNA.ORG_PN=JPEG_SM;DLNA.ORG_OP=01;
DLNA.ORG_CI=1;DLNA.ORG_FLAGS=00f00000000000000000000000000000">http://172.16.149.42:9000/disk/DLNA-PNJPEG_SM-
OP01-CI1-FLAGS00f00000/O0$2$20I523.JPG?scale=360x480</res>
        <res resolution="576x768" protocolInfo="http-get:*:image/jpeg:DLNA.ORG_PN=JPEG_MED;DLNA.ORG_OP=01;
DLNA.ORG_CI=1;DLNA.ORG_FLAGS=00f00000000000000000000000000000">http://172.16.149.42:9000/disk/DLNA-
PNJPEG_MED-OP01-CI1-FLAGS00f00000/O0$2$20I523.JPG?scale=576x768</res>
        <res resolution="810x1080" protocolInfo="http-get:*:image/jpeg:DLNA.ORG_PN=JPEG_LRG;DLNA.ORG_OP=01;
DLNA.ORG_CI=1;DLNA.ORG_FLAGS=00f00000000000000000000000000000">http://172.16.149.42:9000/disk/DLNA-
PNJPEG_LRG-OP01-CI1-FLAGS00f00000/O0$2$20I523.JPG?scale=810x1080</res>
        <class>object.item.imageItem.photo</class>
</meta>
```

### An RSS <meta> XML element (Video)

```
<meta id="0$3$35$2245R524" refID="0$3$27I524" parentID="0$3$35$2245" restricted="1">
        <duration>0:01:53.491</duration>
        <title>Alexander Trailer</title>
        <date>2011-11-18</date>
        <genre>Unknown</genre>
        <album>Alt</album>
        <albumArtURI profileID="JPEG_TN">http://172.16.149.42:9000/disk/DLNA-PNJPEG_TN-CI1-FLAGS00f00000
/defaultalbumart/nocover_video.jpg/O0$3$27I524.jpg?scale=160x160</albumArtURI>
        <extension>wmv</extension>
        <rating>1</rating>
        <modificationTime>1321621594</modificationTime>
        <addedTime>1322752206</addedTime>
        <lastUpdated>1321621594</lastUpdated>
        <res duration="0:01:53.491" size="132289320" resolution="1920x1080" bitrate="9600" protocolInfo="
http-get:*:video/x-ms-wmv:DLNA.ORG_PN=WMVHIGH_PRO;DLNA.ORG_OP=11;DLNA.
ORG_FLAGS=01700000000000000000000000000000" timeseekinfo="http://172.16.149.42:9000/disk/O0$3$35$2245R524.
seek">http://172.16.149.42:9000/disk/DLNA-PNWMVHIGH_PRO-OP11-FLAGS01700000/O0$3$27I524.wmv</res>
        <res bitrate="9600" protocolInfo="http-get:*:image/jpeg:DLNA.ORG_PN=JPEG_TN;DLNA.
ORG_FLAGS=00f00000000000000000000000000000" timeseekinfo="http://172.16.149.42:9000/disk/O0$3$35$2245R524.
seek">http://172.16.149.42:9000/disk/DLNA-PNJPEG_TN-CI1-FLAGS00f00000/defaultalbumart/nocover_video.jpg
/O0$3$27I524.jpg?scale=160x160</res>
        <class>object.item.videoItem.movie</class>
</meta>
```

Syntax:

```
/nmc/rpc/add_metadata?renderer=[BOOKMARK]&url=[url]&metadata=[data]&index=[value]
```

URL:

```
/nmc/rpc/add_metadata?renderer=uuid:56076f6e-6b79-4d65-642b-842b2ba76114&url=http://192.168.100.100:9000/disk
/DLNA-PNMP3-OP01-FLAGS01700000/O0$1$8I6666.mp3&metadata=%3Citem%20id=%22O0$1$8I6666%22%20parentID=%22O0$1$8%
22%20restricted=%221%22%3E%3Cdc:title%3EMP3-L-01%3C/dc:title%3E%3Cdc:date%3E2010-01-01%3C/dc:date%3E%3Cupnp:
genre%3EOther%3C/upnp:genre%3E%3Cupnp:album%3EAP%20Collection%20Vol%204%3C/upnp:album%3E%3Cdc:creator%
3EGeoSuPhat%3C/dc:creator%3E%3Cupnp:albumArtURI%20dlna:profileID=%22JPEG_TN%22%20%3Ehttp://192.168.100.100:
9000/disk/defaultalbumart/nocover_audio.jpg/OO0$1$8I6666.jpg?scale=160x160%3C/upnp:albumArtURI%3E%3Cupnp:
artist%3EGeoSuPhat%3C/upnp:artist%3E%3Cpv:extension%3Emp3%3C/pv:extension%3E%3Cupnp:albumArtist%3EGeoSuPhat%
3C/upnp:albumArtist%3E%3Cpv:modificationTime%3E1285172443%3C/pv:modificationTime%3E%3Cpv:addedTime%
3E1318801106%3C/pv:addedTime%3E%3Cpv:lastUpdated%3E1285172443%3C/pv:lastUpdated%3E%3Cres%20duration=%221:54:
00%22%20size=%22273613952%22%20bitrate=%22320%22%20protocolInfo=%22http-get:*:audio/mpeg:DLNA.ORG_PN=MP3;
DLNA.ORG_OP=01;DLNA.ORG_FLAGS=01700000000000000000000000000000%22%20%3Ehttp://192.168.100.100:9000/disk/DLNA-
PNMP3-OP01-FLAGS01700000/OO0$1$8I6666.mp3%3C/res%3E%3Cupnp:class%3Eobject.item.audioItem.musicTrack%3C/upnp:
class%3E%3C/item%3E
```

Function returns:

```
"ok"
```

The song was added successfully.

### Adding An Unsupported Item To A Renderer Queue

A picture shall be added to the queue. Since the SoundBridge is a music player and cannot show pictures, the call will return an error message.

Syntax:

```
can play?renderer=[BOOKMARK]&item=[BOOKMARK]
```

URL:

```
/nmc/rpc/can_play?renderer=uuid%3A526F6B75-536F-756E-6442-000D4B3063D2&item=uuid:55076f6e-6b79-4d65-646d-
000129d7de2d,0$2$25$63R1290,1,0,Root,0,,0,0,Photos,0$2,,0,0,Album,0$2$25,,3,0,04-08-London,0$2$25$63,,\|5
```

Function returns:

```
1
```

### A second song is added to the end of the queue of the SoundBridge

Syntax:

```
add_bookmark?renderer=[BOOKMARK]&item=[BOOKMARK]&index=[value]
```

URL:

```
/nmc/rpc/add_bookmark?renderer=uuid%3A526F6B75-536F-756E-6442-000D4B3063D2&item=uuid:55076f6e-6b79-4d65-646d-000129d7de2d,0$1$12$324R53514,0,0,Root,0,,0,0,Music,0$1,,20,0,Album,0$1$12,,1,0,Reload,0$1$12$324,,\|5&index=1
```

Function returns:

```
"ok"
```

### The second song shall be made the first item in the queue

Syntax:

```
move_to?renderer=[BOOKMARK]&index=[value]&toindex=[value]
```

URL:

```
/nmc/rpc/move_to?renderer=uuid%3A526F6B75-536F-756E-6442-000D4B3063D2&index=1&toindex=0
```

Function returns:

```
""
```

### Play the songs in the queue.

Syntax:

```
play?renderer=[BOOKMARK]&mode=0
```

URL:

```
/nmc/rpc/play?renderer=uuid%3A526F6B75-536F-756E-6442-000D4B3063D2&mode=0
```

Function returns:

```
""
```

### Get the play state.

Syntax:

```
get_state?renderer=[BOOKMARK]
```

URL:

```
/nmc/rpc/get_state?renderer=uuid%3A526F6B75-536F-756E-6442-000D4B3063D2
```

Function returns:

```
"1|0:03:08|0:04:39"
```

State is 1, that means the song is playing. The current position is 0:03:08 and the duration of the song is 0:04:39.

### Get all server and renderer events.

Syntax:

```
get_events?eventid=[last event ID]
```

URL:

```
/nmc/rpc/get_events?eventid=0
```

Function returns:

```
1 server 4 uuid:55076f6e-6b79-4d65-646d-000129d7de2d,0|2 server 1 uuid:55076f6e-6b79-4d65-646d-000129d7de2d,
0|3 server 1 uuid:55076f6e-6b79-4d65-646d-000129d7de2d,0|4 renderer 16386 uuid:526F6B75-536F-756E-6442-
000D4B3063D2
```

The returned string in a more readable format:

| Event ID | server/renderer | Code | Bookmark |
|---|---|---|---|
| 1 | server | 4 | uuid:55076f6e-6b79-4d65-646d-000129d7de2d,0 |
| 2 | server | 1 | uuid:55076f6e-6b79-4d65-646d-000129d7de2d,0 |
| 3 | server | 1 | uuid:55076f6e-6b79-4d65-646d-000129d7de2d,0 |
| 4 | renderer | 16386 | uuid:526F6B75-536F-756E-6442-000D4B3063D2 |

### Server and renderer events

| Server codes | What they mean |
|---|---|
| 1 | server updated |
| 2 | lost connection to server |
| 4 | server detected |

| Renderer codes | What they mean |
|---|---|
| 0x4001 (16385) | lost contact to renderer |
| 0x4002 (16386) | renderer detected |

### Error return codes

invalid args given for this rpc call
error on creating a context
error on deleting the context
bookmark not found
execution of the action failed
unknown error

## Logging source and level

The log source specifies the module that should be logged. It is an OR'ed combination of these values:

### Logging Sources

| Source number | Source name |
|---|---|
| 1 | Common system messages |
| 2 | Device discovery (generates lots of output!) |
| 8 | HTTP communication |
| 32 | Device events and subscription messages |
| 64 | Queue handler communication |
| 128 | High level API |
| 4096 | DTCP-IP |

The log source could now be set to 1+8+32=41 to enable logging for system, http and event messages. The default is zero which means logging is disabled. Other values are used by Twonky server and are ignored by the client.

### Logging Levels

| Level number | Level name |
|---|---|
| 1 | Trace |
| 2 | Info |
| 3 | Warning |
| 4 | Error |
| 5 | Critical |

A lower number includes logs for all higher numbers. The lower the number the more is logged.
Default is 4.

Example: A value of 9 for **source** and 2 for **level** enables info, warning, error and critical logs for the system and HTTP modules.

# Search Syntax

## General Search Syntax

`http://127.0.0.1:9000/nmc/rpc/search?server=[BOOKMARK]search=[hex-encoded query]start=[value]count=[value]`

Performs a search for the specified query. The search parameter is better described below.
Note: The search query must be hex-encoded. For paging through a search, the start and count parameters can be used.

The search syntax takes two forms.

One can use the UPnP search syntax:
upnp:class derivedfrom "object.item.audioItem.musicTrack" and (dc:title contains "FindThis" or upnp:genre contains "FindThis" or upnp:artist contains "FindThis")

Once hex-encoded, this can be used with a search RPC call as follows:

```
http://127.0.0.1:9000/nmc/rpc/search?server=uuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,
0&search=75706e703a636c617373206465726976656466726f6d20226f626a6563742e6974656d2e617564696f4974656d2e6d75736963
547261636b2220616e64202864633a7469746c6520636f6e7461696e73202246696e645468697322206f722075706e703a67656e726520
36f6e7461696e73202246696e645468697322206f722075706e703a6172746973742063f6e7461696e73202246696e645468697322229&s
tart=0&count=10
```

One can also use a simplified search syntax:
type=musicItem&artist=FindThis&album=FindThat

This also must be hex-encoded if used in the RPC, and would look like:

```
http://127.0.0.1:9000/nmc/rpc/search?server=uuid%3A4f1bf61c-599b-443d-b41d-36408362ec75,
0&search=747970653d6d757369634974656d266172746973743d46696e645468697326616c62756d3d46696e6454686174&start=0&cou
nt=10
```

## Simplified Search Syntax

The simpler search syntax is presented below and uses URL-encoded parameters to set up the search. Caveat: These search tokens are combined into an AND search, e.g. a searched-for item must have all of the token values in its metadata before it matches. There is currently no provision for OR searches.

By default, these searches will use the 'contains' search semantic, meaning that they will match substrings in the metadata. e.g. 'title=FindThis' will match when a title is 'FindThisThing'.
To force an exact, whole-word match of a metadata string, append the 'exact=1' parameter somewhere in the search string. e.g. 'title=FindThis&exact=1' will not match when a title is 'FindThisThing'.

It will only match when a title is 'FindThis'.

The following parameters are allowed in a **URL-encoded** query string format.

keyword - value
title - the title
creator - the creator
genre - the genre
album - the album
description - description
date - date
class - media item's class
author - the author
actor - the actor
director - director
albumArtist - album artist
trackNumber - track number
seriesTitle - series title
seriesID - series ID
episodeNumber - episode number
episodeCount - episode count
channelNr - channel number
channelName - channel name
longDescription - long description
rating - upnp:rating
resolution - resolution
duration - duration
id - @id
refID - @refID
protocolInfo - protocol info
userRating - pv:rating

type - the type
The predefined "type" values are:
musicItem, musicAlbum, musicArtist, musicGenre,
photoItem, photoAlbum,
videoItem,
playlist,
folder,
container,
item

### Examples

Find all music tracks that contain FindThis in the title.

```
type=musicItem&title=FindThis
```

Find all music tracks in the FindThis album.

```
type=musicItem&album=FindThis
```

Find all music tracks with FindThis as the artist, and FindThat as the album.

```
type=musicItem&artist=FindThis&album=FindThat
```

Find all music albums with FindThis as the genre.

```
type=musicAlbum&genre=FindThis
```

URL encoding of special characters searching for music item.

Actual title: a"&=b

URL encoded string: title=a%22%26%3db&type=musicItem

## UPnP Search Syntax

The UPnP Search Syntax grammar is as follows:

```
searchCrit ::= searchExp | asterisk
searchExp ::= relExp | searchExp wChar+ logOp wChar+ searchExp | '(' wChar* searchExp wChar* ')'
logOp ::= 'and' | 'or'
relExp ::= property wChar+ binOp wChar+ quotedVal | property wChar+ existsOp wChar+ boolVal
binOp ::= relOp | stringOp
relOp ::= '=' | '!=' | '<' | '<=' | '>' | '>='
stringOp ::= 'contains' | 'doesNotContain' | 'derivedfrom'
existsOp ::= 'exists'
boolVal ::= 'true' | 'false'
quotedVal ::= dQuote escapedQuote dQuote
wChar ::= space | hTab
```

### Examples

Find all music tracks that contain FindThis in the title.

```
upnp:class derivedfrom "object.item.audioItem.musicTrack" and (dc:title contains "FindThis")
```

Find all music tracks in the FindThis album.

```
upnp:class derivedfrom "object.item.audioItem.musicTrack" and (upnp:album contains "FindThis")
```

Find all music tracks with FindThis as the artist, and FindThat as the album.

```
upnp:class derivedfrom "object.item.audioItem.musicTrack" and (upnp:artist contains "FindThis")
```

Find all music albums with FindThis as the genre.

```
upnp:class derivedfrom "object.item.audioItem.musicTrack" and (upnp:genre contains "FindThis")
```

### Exact Searches

Find a music track with "FindThis" as the title.

```
upnp:class derivedfrom "object.item.audioItem.musicTrack" and (dc:title = "FindThis")
```

### Searchable Fields

Items can be searched for by specifying some combination of the following UPnP fields.

```
res@resolution
res@duration
dc:title
dc:creator
upnp:actor
upnp:artist
upnp:genre
upnp:album
dc:date
upnp:class
@id
@refID
@protocolInfo
upnp:author
dc:description
pv:avKeywords
pv:rating
upnp:seriesTitle
upnp:episodeNumber
upnp:director
upnp:rating
upnp:channelNr
upnp:channelName
upnp:longDescription
pv:capturedate
pv:custom
```

### derivedfrom

derivedfrom can also be one of the following:

```
object.item
object.item.audioItem.musicTrack
object.item.audioItem.audioBroadcast
object.item.audioItem.online.musicTrack
object.item.imageItem.photo
object.item.imageItem.online.photo
object.item.videoItem.movie
object.item.videoItem.classified.movie
object.item.videoItem.videoBroadcast
object.item.videoItem.online.movie
object.container
object.container.playlistContainer
object.container.musicContainer
object.container.pictureContainer
object.container.videoContainer
object.container.storageFolder
object.container.album.musicAlbum
object.container.album.photoAlbum
object.container.person.musicArtist
object.container.genre.musicGenre
```

## DTCP Support

If the REST API is used with an DTCP-IP enabled Twonky Server, then support for moving or copying DTCP protected content to other DTCP enabled media servers ("DTCP push move") is available.
To help determine if a Twonky Server is DTCP enabled and is supporting this feature, the UPnP device description XML of the server contains the following tags:
<pv:extension xmlns:pv="http://www.pv.com/pvns/">dtcp-push</pv:extension>

To determine if an intended target media server supports this feature, the UPnP device description XML of this server must expose the following tags:
<dlna:X_DLNACAP xmlns:dlna="urn:schemas-dlna-org:device-1-0">av-upload,image-upload,audio-upload,**dtcp-copy,dtcp-move**</dlna:X_DLNACAP>

These tags and data can be accessed through the web API of the Client SDK through the metadata of a selected media server.

Please see "dtcp_content_upload" and "dtcp_get_capabilities" in the Client SDK RPC for further details on the "DTCP push move" feature.

#### Document References


1. Twonky Client SDK Introduction
2. Twonky Server, Technical Specification and APIs
3. UPnP Device Architecture 1.1, www.upnp.org
4. UPnP Media Server:1 and Media Renderer:1, www.upnp.org
5. DLNA Design Guidelines August 2009, www.dlna.org
6. UPnP Developer Tools, http://opentools.homeip.net/dev-tools-for-upnp


#### Glossary

**DMS, media server, DLNA MS, UPnP Media Server Device**
A UPnP and or DLNA specified media server that implements the UPnP Media Server device specification.
See www.upnp.org and www.dlna.org/industry for further details

**DMP, networked media player Device, UPnP Media Player Device**
A networked media player device that implements the UPnP Media Player Device specification. A DMP will need a user interface to enable a user to control it to find and select content from a DMS in his network (see also DMR).
*See www.upnp.org and www.dlna.org/industry for further details*

**DMR networked media renderer Device, UPnP AV CP, DLNA MSCP & MRCP**
A networked Media Renderer Device that implements a networked media player and offering to be remotely controlled from a DMC. A DMR can be co-located with a DMP or exist „head-less" for remote control only („network media adapter").
*See www.upnp.org and www.dlna.org/industry for further details*

**DMC, UPnP AV CP, DLNA MSCP & MRCP**
A networked remote control application used for discovering and controlling media renderers and media servers in a network.
*See www.upnp.org and www.dlna.org/industry for further details*

**DTCP push move**
The Web APIs allow to move DTCP-IP protected content to other DTCP-IP enabled media servers.

**NMC, Networked Media Controller**

Originally specified the high-level API of the Twonky Client SDK. This is going to be cleared, but to keep backwards compatibility with existing applications the abbreviation is kept in the URLs.