DVBLink Client API

Version 0.9.4
8 June 2017

Introduction

DVBLink Client API is a set of function calls that allows building a custom DVBLink client. The API uses http transactions with xml-structured parameters.
As a prerequisite DVBLink Client API requires installation of DVBLink Server (DVBlink Connect! Server for Windows) on the server system.

Supported stream types

Transcoding
DVBLink server supports video transcoding – changing video and audio parameters of the stream. The overview of the supported transcoding features and platforms can be found at the following link:
http://dvblogic.com/wiki/index.php/How_to_setup_and_use_video_transcoding_feature_of_the_DVBLink_server
Some of the stream types are only available on platforms that support stream transcoding. In the list of supported stream types below, those types that are marked with *.

Live TV
The following stream types are supported for streaming live TV:

- Raw transport stream via UDP
- *HLS (IPhone-type stream)
- ~~ASF (windows media stream)~~ *(not supported anymore since DVBLink v5.0)*
- *Transport stream with mpeg-4 video and AAC audio over RTSP (android)
- Raw transport stream via HTTP
- Raw transport stream via HTTP with server side time-shifting capabilities
- *Transport stream with h264 video and AAC audio via HTTP
- *Mp4 segmented stream via HTTP
- *Webm stream via HTTP

Playback objects (Recorded TV)
The following stream types are supported for streaming recorded TV:

- *HLS (IPhone-type stream)
- Raw transport stream via HTTP

Direct vs Indirect http streaming

Raw transport stream via HTTP can be request either directly or indirectly.
Indirect stream request works with a request handle and uses *play_channel* API function and will be discussed later in this document.

Direct request for raw transport stream via HTTP

!!! Recommended way of getting direct streaming URL for a channel is to use **get_channel_url** function !!

Direct HTTP stream can be requested via url of the following format:

http://**<server ip>**:**<streaming server port + 1>**/dvblink/direct?client=**<client_id>**&channel=**<channel_id>**

where

- <server ip> - IP address of DVBLink server
- <server port> - base **streaming** port of DVBLink server (default – 8100)
- <client_id> - a unique id of the client that requests a stream. Can be any string of alphanumeric characters without spaces as long as it is unique – a guid for example. *If empty or not present, client id will be generated from the client's IP address.*
- <channel_id> - DVBLink channel id. You can find it under <frequency> tag in

dvblink_channel_storage.xml for each channel. It is a long number – something like 10010000

For example:
http://192.168.0.100:8101/dvblink/direct?client=AAABBBCCC&channel=10460000

Direct request for transcoded stream via HTTP

Transcoded stream can be requested via url of the following format:

http://**<server ip>**:**<streaming server port + 1>**/dvblink/direct?client=**<client_id>**&channel=**<channel_id>**
&transcoder=**<format>**&height=**<height>**&width=**<width>**&bitrate=**<bitrate>**&lng=**<lng_code>**

where
- <server ip> - IP address of DVBLink server
- <server port> - base **streaming** port of DVBLink server (default – 8100)
- <client_id> - a unique id of the client that requests a stream. Can be any string of alphanumeric characters without spaces as long as it is unique – a guid for example. *If empty or not present, client id will be generated from the client's IP address.*
- <channel_id> - DVBLink channel id. You can find it under <frequency> tag in dvblink_channel_storage.xml for each channel. It is a long number – something like 10010000
- <format> - format of the output stream: mp4, webm, h264ts
- <height> - Frame height of the transcoded stream
- <width> - Frame width of the transcoded stream
- <bitrate> - Bitrate in kbits/sec of the transcoded stream
- <lng> - (optional) an ISO 639 code of the language

For example:
http://192.168.0.100:8101/dvblink/direct?client=AAABBBCCC&channel=10460000&transcoder=mp4&height=480&width=640&bitrate=512&lng=eng

Authentication

DVBLink Connect! Server requires basic HTTP authentication with user name and password.

Generic request format

Request has to be sent using POST command to the following http address:
http://**<server ip>**:**<server base streaming port>**/cs/

Where
- <server ip> - IP address of DVBLink server
- <server base streaming port> - base streaming port of DVBLink server (default 8100)

The following parameter string has to be POSTed:
command**=<dvblink command>**&xml_param=**<xml data>**

Where
- <dvblink command> - function to perform (see below for the list of functions)
- <xml data> - function dependent set of parameters in xml format

Generic response format

Response is sent in xml format with the following generic structure:

```
<response xmlns="http://www.dvblogic.com">
        <status_code/>
        <xml_result/>
</response>
```

Where
- <status_code/> - mandatory field of integer type. Can have the following values:

    - STATUS_OK = 0
    - STATUS_ERROR = 1000
    - STATUS_INVALID_DATA = 1001
    - STATUS_INVALID_PARAM = 1002
    - STATUS_NOT_IMPLEMENTED = 1003
    - STATUS_MC_NOT_RUNNING = 1005
    - STATUS_NO_DEFAULT_RECORDER = 1006
    - STATUS_MCE_CONNECTION_ERROR = 1008
- <xml_result/> - contains function specific result information (optional)

Content-Type and encoding

POST requests should pass parameters in the body in a URL-encoded form, and use Content-Type **application/x-www-form-urlencoded**

Response is sent back using **UTF-8** encoding.

Date/time type

All date/time parameters, used in xml structures, have *long* type. They are number of seconds, counted from UNIX epoc: 00:00:00 UTC on 1 January 1970.
All duration parameters have *long* type and are expressed in seconds.

Bool type

If xml request or response format contains a field of bool type and thios field is not present in the actual response/request then its value by default is considered to be False.
If the value is present then it can be either False or True.

Schedules and recordings

DVBLink recording functionality operates in terms of schedules and recordings.
Schedule defines a rule what has to be recorded. The following examples can be considered: “Record all series of a particular EPG program” or “Record a program on a channel 12 every day from 14:00 until 15:00”.
Each schedule produces a number (zero or more) of *recordings* – these are actual tasks to do the recording. The second example above will produce a recording for each day on channel 12 starting at 14:00 until 15:00.

Playback objects

The recorded TV files and other streams with a limited (known) duration – as opposed to live TV streams, which have infinite duration – are presented by DVBLink server as playback objects.
Playback objects are organized in a hierarchy of playback items and playback containers. If analogy to a file system can be made then items are files and containers are folders.
Every playback object has a unique id. The hierarchy begins at DVBLink server container, which is the top level parent of all objects and has empty string as its id.
Playback items can be of several types – namely recorded TV, video, images and music. Each playback item type has its own metadata set.

Broadcast standards

Whenever applicable, the following values for broadcast standards are used:
unknown (0x00000000), dvb-t (0x00000001), dvb-c (0x00000002), dvb-s (0x00000004), atsc (0x00000008), clearqam (0x00000010), iptv (0x00000020)

DVBLink commands

DVBLink Connect! Server implements the following commands (functions, marked as provisional, are not implemented yet)

1. get_channels
2. get_all_channels (provisional)
3. get_favorites

4. play_channel
5. stop_channel
6. search_epg
7. get_recordings
8. add_schedule
9. remove_schedule
10. remove_recording
11. set_parental_lock
12. get_parental_status
13. get_schedules
14. update_schedule
15. get_playlist_m3u
16. get_object
17. remove_object
18. stop_recording
19. get_streaming_capabilities
20. get_recording_settings
21. set_recording_settings
22. get_server_info
23. execute_command
24. get_channel_url
25. timeshift_get_stats
26. timeshift_seek
27. get_devices (provisional)
28. get_scanners (provisional)
29. start_scan (provisional)
30. cancel_scan (provisional)
31. apply_scan (provisional)
32. get_networks (provisional)
33. get_device_status (provisional)
34. repair_database (provisional)
35. force_epg_update (provisional)
36. get_channels_visibility (provisional)
37. set_channels_visibility (provisional)
38. get_epg_sources (provisional)
39. get_epg_channels (provisional)
40. get_epg_channel_config (provisional)
41. set_epg_channel_config (provisional)
42. match_epg_channels (provisional)
43. get_installed_products (provisional)
44. activate_product (provisional)
45. activate_product_trial (provisional)
46. updater_get_status (provisional)
47. updater_check_update (provisional)
48. updater_start_update (provisional)

Function get_channels

DVBLink command
get_channels

Request xml data
```
<?xml version="1.0" encoding="utf-8" ?>
<channels xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com" />
```

Response xml_result
```
<?xml version="1.0" encoding="utf-8" ?>
<channels xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
  <channel>
    <channel_id/> - string mandatory
```

<channel_dvblink_id/> *- long mandatory*
<channel_name/> *- string mandatory*
<channel_number/> *- int optional*
<channel_subnumber/> *- int optional*
<channel_type> *- int mandatory, (RD_CHANNEL_TV = 0, RD_CHANNEL_RADIO = 1, RD_CHANNEL_OTHER = 2)*
<channel_child_lock> *- bool optional*
<channel_logo/> *- string optional, url to channel logo*
<channel_encrypted/> *- int optional,1 if encrypted, 0 if FTA (default)*
</channel>
…
</channels>

Notes:

- <channel_id> is a generic channel identifier used in all other commands that perform operations on the channel(s). The exception is *play_channel* command that uses <channel_dvblink_id> to play channel.

Function get_all_channels (provisional)

DVBLink command
get_all_channels

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<channels xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com" />

Response xml_result
Response result format is identical to what get_channels function returns.

Notes:
This functions returns all available channels at the server with their original metadata (thus ignoring channel visibility settings and channel name/number overrides)

Function get_favorites

DVBLink command
get_favorites

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<channels xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com" />

Response xml_result
<?xml version="1.0" encoding="utf-8" ?>
<favorites xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
<favorite>
<id/> *- string mandatory*
<name/> *- string mandatory*
<channels>
<channel/> *- string mandatory, refers to channel_id in get_channels call*
…
</channels>
</favorite>
…
</favorites>

Function play_channel

DVBLink command
play_channel

Request xml data

```
<?xml version="1.0" encoding="utf-8" ?>
<stream xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com">
        <channel_dvblink_id/> - long mandatory
        <client_id/> - string mandatory
        <stream_type/> - string mandatory (rtp, hls, asf, raw_http, raw_udp)
        <server_address/> - string mandatory
        <client_address/> - string optopnal (used for raw_udp)
        <streaming_port/> - ushort optinal (used for raw_udp)
        <transcoder> - optional
                <height/> - uint mandatory
                <width/> - uint mandatory
                <bitrate/> - uint optional
                <audio_track/> - string optional
        </transcoder>
        <duration/> - long optional
</stream>
```

Notes:

- <channel_dvblink_id> is a DVBLink specific channel identifier
- <client_id> is the unique identification string of the client. Should be the same across all DVBLink Client API calls from a given client. Can be a uuid for example or id/mac of the client device
- <stream_type> is the type of requested stream. The following values are supported: “rtp”, “hls”, “raw_http”, “raw_udp”, “webm”, “mp4”, “h264ts”, “raw_http_timeshift”, “h264ts_http_timeshift”
- <server_address> - ip address/server network name of the DVBLink server
- <transcoder> is optional element, defining the parameters of transcoded stream. It is used for certain stream types, namely rtp, hls and asf.
    - <height> is height in pixels of the transcoded video stream
    - <width> is width in pixels of the transcoded video stream
    - <bitrate> is bitrate in kilobits/sec of the transcoded video stream
    - <audio_track> is ISO-639 language code that is used to choose a particular audio track for transcoding
- <duration> is the timeout until channel playback is stopped by a server (indefinite if not specified)
- If <stream_type> is not “raw_http”, “raw_udp” or “raw_http_timeshift” then <transcoder> element must be present

Response xml_result

```
<?xml version="1.0" encoding="utf-8" ?>
<stream xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
        <channel_handle/> - long mandatory
        <url/> - string mandatory
</stream>
```

Notes

- <channel_handle> is the channel handle of a playing channel. It should be used in *stop_channel* command to stop stream. Attention, if stream is not stopped server will play it indefinitely (unless it has a client disconnect detection mechanism for a particular protocol).
- <url> is the uri that client can use to read the stream.
- There can be more than one stream started for the same <client_id>. Each stream will have its own handle and should be stopped separately. Alternatively client can stop all its streams at once using <client_id> parameter for *stop_channel* command.
- Server has client disconnect detection functionality implemented for all protocols except udp-based

Function stop_channel

DVBLink command
stop_channel

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<stop_stream xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com">
<channel_handle/> *- long mandatory*
or
<client_id/> *- string mandatory*
</stop_stream>

Notes
- Function stop channel playback for a particular channel handle (if <channel_handle> is supplied as a parameter) or for all client's streams (if <client_id> is supplied as a parameter)

Response xml_result
Response contains only <status_code>

Function search_epg

DVBLink command
search_epg

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<epg_searcher xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<channels_ids> *- optional*
<channel_id/> *- string optional*
...
</channels_ids>
<program_id/> *– string optional*
<keywords/> *- string optional*
<genre_mask/> *- unsigned int optional*
<requested_count/> *- int optional*
<start_time/> *– mandatory long*
<end_time/> *– mandatory long*

<epg_short/> *– optional bool (false by default)*
</epg_searcher>

Notes
- Search can be done on the following criteria:
  - Channel ids: returns all programs for the specified channels. If no channels are specified, search request returns matches for all channels.
  - Program id: returns information about a particular program
  - Keywords (or better to say a keyphrase): returns all programs that have the provided keyphrase in their title or short description
  - Genre mask: returns all programs that match one of the specified genres
  - Start/end time: returns all programs that fully or partially fall into a specified time span. One of the parameters (<start_time> and <end_time> may be -1, meaning that search is not bound on that side)
- Search is case-insensitive and ignores all non-alpha-numeric characters
- If search keyphrase is augmented with double quotes – e.g. "pointless" – then exact phrase match will be searched. If search keyphrase starts with hash – e.g. #pointless – then only program title will be search. The two search specifiers may be combined.
- Some of the search criteria can be combined:
  - Program id search is always exclusive to keyphrase, genre mask and start/end time

searches
    - Search results are combined on AND principles
- <epg_short> flag specifies the level of returned EPG information (allows for faster EPG overview build up). Short EPG information includes only program id, name, start time, duration, genre info, premiere, repeat flags and record/record series/record conflict flags.
- Number of returned programs is limited by <requested_count/>. If it is set to -1 or absent then no limit is set on a number of returned programs.

Genre mask has the following values for genres:

RDGC_ANY = 0x00000000,
RDGC_NEWS = 0x00000001,
RDGC_KIDS = 0x00000002,
RDGC_MOVIE = 0x00000004,
RDGC_SPORT = 0x00000008,
RDGC_DOCUMENTARY = 0x00000010,
RDGC_ACTION = 0x00000020,
RDGC_COMEDY = 0x00000040,
RDGC_DRAMA = 0x00000080,
RDGC_EDU = 0x00000100,
RDGC_HORROR = 0x00000200,
RDGC_MUSIC = 0x00000400,
RDGC_REALITY = 0x00000800,
RDGC_ROMANCE = 0x00001000,
RDGC_SCIFI = 0x00002000,
RDGC_SERIAL = 0x00004000,
RDGC_SOAP = 0x00008000,
RDGC_SPECIAL = 0x00010000,
RDGC_THRILLER = 0x00020000,
RDGC_ADULT = 0x00040000

Response xml_result

```
<?xml version="1.0" encoding="utf-8" ?>
<epg_searcher xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
        <channel_epg> - optional
                <channel_id/> - string mandatory
                <dvblink_epg> - mandatory
                        <program> - mandatory
                                <program_id – string mandatory
                                <name/> — string mandatory
                                <start_time/> — long mandatory
                                <duration/> — long mandatory

                                string optional:
                                <short_desc/>
                                <subname/>
                                <language/>
                                <actors/>
                                <directors/>
                                <writers/>
                                <producers/>
                                <guests/>
                                <categories/>
                                <image/>

                                long optional:
                                <year/>
                                <episode_num/>
                                <season_num/>
                                <stars_num/>
                                <starsmax_num/>

                                Optional.true if tag is present, false otherwise:
```

```
                        <hdtv/>
                        <premiere/>
                        <repeat/>
                        <cat_action/>
                        <cat_comedy/>
                        <cat_documentary/>
                        <cat_drama/>
                        <cat_educational/>
                        <cat_horror/>
                        <cat_kids/>
                        <cat_movie/>
                        <cat_music/>
                        <cat_news/>
                        <cat_reality/>
                        <cat_romance/>
                        <cat_scifi/>
                        <cat_serial/>
                        <cat_soap/>
                        <cat_special/>
                        <cat_sports/>
                        <cat_thriller/>
                        <cat_adult/>
                        <is_record/>
                        <is_series/>
                        <is_repeat_record/>
                        <is_record_conflict/>
                    </program>
                    ...
                </dvblink_epg>
            </channel_epg>
            …
</epg_searcher>
```

Notes

- <is_record/> flag indicates if this program is scheduled for recording
- <is_repeat_record/> flag indicates if this program is part of record series schedule
- <is_record_conflict/> flag indicates that this program is scheduled for recording and is conflicting with another recording timer
- <is_series/> flag indicates if this program can be scheduled for series recording
- If for a particular program <is_repeat_record/> is set and <is_record/> is not set, this means that a recording timer for this program has been cancelled

Function add_schedule

DVBLink command
add_schedule

Request xml data

```
<schedule xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com">
        <user_param/> - string optional
        <force_add/> - bool optional
        <margine_before/> - int optional, margin value in seconds, if -1 then default margin is used
        <margine_after/> - int optional, margin value in seconds, if -1 then default margin is used

        <by_epg> - mandatory
            <channel_id/> - string mandatory
            <program_id/> - string mandatory
            <repeat/> - bool optional
            <new_only/> - bool optional, indicates that only new programs have to be recorded
            <record_series_anytime/> - bool optional, indicates whether to record only series starting
            around original program start time or any of them
            <recordings_to_keep/> - int mandatory for series recording (1,2,3,4,5,6,7,10; 0 – keep all)
```

</by_epg>
or
<manual> *- mandatory*
<channel_id/> *- string mandatory*
<title/> *- string optional*
<start_time/> *- long mandatory*
<duration/> *- long mandatory*
<day_mask/> *— long mandatory (DAY_MASK_SUN = 1,*
DAY_MASK_MON = 2,
DAY_MASK_TUE = 4,
DAY_MASK_WED = 8,
DAY_MASK_THU = 16,
DAY_MASK_FRI = 32,
DAY_MASK_SAT = 64,
DAY_MASK_DAILY = 255)
<recordings_to_keep/> *- int mandatory for repeated recordings (1,2,3,4,5,6,7,10; 0 – keep all)*
</manual>
or
<by_pattern> *- mandatory*
<channel_id/> *- string mandatory*
<key_phrase/> *- string optional (either key_phrase or genre_mask or both should be present)*
<genre_mask/> *— long optional (Genre mask has the following values for genres:*
RDGC_ANY = 0x00000000,
RDGC_NEWS = 0x00000001,
RDGC_KIDS = 0x00000002,
RDGC_MOVIE = 0x00000004,
RDGC_SPORT = 0x00000008,
RDGC_DOCUMENTARY = 0x00000010,
RDGC_ACTION = 0x00000020,
RDGC_COMEDY = 0x00000040,
RDGC_DRAMA = 0x00000080,
RDGC_EDU = 0x00000100,
RDGC_HORROR = 0x00000200,
RDGC_MUSIC = 0x00000400,
RDGC_REALITY = 0x00000800,
RDGC_ROMANCE = 0x00001000,
RDGC_SCIFI = 0x00002000,
RDGC_SERIAL = 0x00004000,
RDGC_SOAP = 0x00008000,
RDGC_SPECIAL = 0x00010000,
RDGC_THRILLER = 0x00020000,
RDGC_ADULT = 0x00040000)
<recordings_to_keep/> *- int mandatory for repeated recordings (1,2,3,4,5,6,7,10; 0 – keep all)*
</by_pattern>
</schedule>

Notes

- New schedules can be added either manually, based on a certain EPG program or on a search pattern (<manual>, <by_epg> or <by_pattern>)
- <force_add> flag indicates that new schedule should be added even if there are other conflicting schedules present

Response xml_result
Response contains only status_code.

Function get_schedules

DVBLink command
get_schedules

Request xml data

<schedules_request xmlns="http://www.dvblogic.com" />

Response xml_result

<schedules xmlns="http://www.dvblogic.com">
<schedule>
<schedule_id/> *- string mandatory*
<user_param/> *- string mandatory*
<force_add/> *- bool mandatory, If true this flag indicates that schedule has to be added even if there are timer conflicts.*
<margine_before/> *- int mandatory, margin value in seconds*
<margine_after/> *- int mandatory, margin value in seconds*
<targets>
<target/> *- string optional, send to target id for automatic processing of recordings, which are generated under this schedule*
<target/>
….
</targets>

<by_epg> *- mandatory, exclusive with <manual>, <by_pattern>*
<channel_id/> *- string mandatory*
<program_id/> *- string mandatory, an id of program to record*
<repeat/> *- bool optional, indicates whether to record program series*
<new_only/> *- bool optional, indicates that only new programs have to be recorded*
<record_series_anytime/> *- bool optional, indicates whether to record only series starting around original program start time or any of them*
<recordings_to_keep/> *- int mandatory for series recording (1,2,3,4,5,6,7,10; 0 – keep all)*
<program> *- mandatory, program metadata, which was used to set this schedule*

The same fields as for <program> element in <search_epg> function

</program>
</by_epg>

<manual> *- mandatory, exclusive with <by_epg>, <by_pattern>*
<channel_id/> *- string mandatory*
<title/> *- string mandatory*
<start_time/> *- long mandatory*
<duration/> *- int mandatory*
<day_mask/> *— int mandatory (DAY_MASK_SUN = 1, DAY_MASK_MON = 2, DAY_MASK_TUE = 4, DAY_MASK_WED = 8, DAY_MASK_THU = 16, DAY_MASK_FRI = 32, DAY_MASK_SAT = 64, DAY_MASK_DAILY = 255)*
<recordings_to_keep/> *- int mandatory for repeated recordings (1,2,3,4,5,6,7,10; 0 – keep all)*
</manual>

<by_pattern> *- mandatory, exclusive with <by_epg>, <manual>*
<channel_id/> *- string mandatory*
<key_phrase/> *- string optional (either key_phrase or genre_mask or both should be present)*
<genre_mask/> *— long optional (Genre mask has the following values for genres:*
RDGC_ANY = 0x00000000,
RDGC_NEWS = 0x00000001,
RDGC_KIDS = 0x00000002,
RDGC_MOVIE = 0x00000004,
RDGC_SPORT = 0x00000008,
RDGC_DOCUMENTARY = 0x00000010,
RDGC_ACTION = 0x00000020,
RDGC_COMEDY = 0x00000040,
RDGC_DRAMA = 0x00000080,
RDGC_EDU = 0x00000100,
RDGC_HORROR = 0x00000200,

RDGC_MUSIC = 0x00000400,
RDGC_REALITY = 0x00000800,
RDGC_ROMANCE = 0x00001000,
RDGC_SCIFI = 0x00002000,
RDGC_SERIAL = 0x00004000,
RDGC_SOAP = 0x00008000,
RDGC_SPECIAL = 0x00010000,
RDGC_THRILLER = 0x00020000,
RDGC_ADULT = 0x00040000)

<recordings_to_keep/> *- int mandatory for repeated recordings (1,2,3,4,5,6,7,10; 0 – keep all)*

</by_pattern>

</schedule>

…

</schedules>

Function update_schedule

DVBLink command

update_schedule

Request xml data

<update_schedule xmlns="http://www.dvblogic.com">

<schedule_id/> *- string mandatory*

<new_only/> *- bool mandatory*

<record_series_anytime/> *- bool mandatory*

<recordings_to_keep/> *- int mandatory (1,2,3,4,5,6,7,10; 0 – keep all)*

<margine_before/> *- int optional, margin value in seconds, if -1 then existing margin is kept*

<margine_after/> *- int optional, margin value in seconds, if -1 then existing margin is kept*

<targets>

<target/> *- string optional, send to target id for automatic processing of recordings, which are generated under this schedule*

<target/>

….

</targets>

</update_schedule>

Response xml_result

Response contains only status_code.

Function get_recordings (recording timers)

DVBLink command

get_recordings

Request xml data

<?xml version="1.0" encoding="utf-8" ?>

<recordings xmlns:i="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://www.dvblogic.com"/>

Response xml_result

<recordings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">

<recording>

<recording_id> *- mandatory*

<schedule_id> *- mandatory*

<channel_id> - *mandatory*

<is_active> *- optional*

<program> *- mandatory*

The same fields as for <program> element in <search_epg> function

```
                </program>
        <recording>
</recordings>
```

Function remove_schedule

DVBLink command
remove_schedule

Request xml data
```
<remove_schedule xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <schedule_id/> - string mandatory
</remove_schedule>
```

Response xml_result
Response contains only status_code.

Function remove_recording (recording timer)

DVBLink command
remove_recording

Request xml data
```
<remove_recording xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <recording_id/> - string mandatory
</remove_recording>
```

Response xml_result
Response contains only status_code.

Function set_parental_lock

DVBLink command
set_parental_lock

Request xml data
```
<parental_lock xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <client_id/> string mandatory
        <is_enable/> bool mandatory, true – enable parental lock, false – disable parental lock
        <code/> string mandatory if is_enable = false
</parental_lock>
```

Response xml_result
```
<parental_status xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
        <is_enabled/> - bool mandatory
</parental_status>
```

Function get_parental_status

DVBLink command
get_parental_status

Request xml data

<parental_lock xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<client_id/> *string mandatory*
</parental_lock>

Response xml_result
<parental_status xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
<is_enabled/> *- bool mandatory*
</parental_status>

Function get_playlist_m3u

DVBLink command
get_playlist_m3u

Notes:
This function returns a playlist with direct http links to the channels' urls

Request xml data
<playlist_request xmlns="http://www.dvblogic.com" />

Response is the m3u string:
#EXTM3U
#EXTINF:0,1 – Channel 1
http://...
#EXTINF:0,2 – Channel 2
http://...
…

Function get_object

Predefined identifiers for built-in containers:
- "8F94B459-EFC0-4D91-9B29-EC3D72E92677" – built-in DVBLink recorder
- "E44367A7-6293-4492-8C07-0E551195B99F" – container with recordings sorted by Name
- "F6F08949-2A07-4074-9E9D-423D877270BB" – container with recordings sorted by Date
- "CE482DD8-BC5E-47c3-9072-2554B968F27C" – container with Genres of the recordings
- "0E03FEB8-BD8F-46e7-B3EF-34F6890FB458" – container with Series of the recordings

DVBLink command
get_object

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<object_requester xmlns="http://www.dvblogic.com">
<object_id/> *- string mandatory (empty string – root object)*
<object_type/> *- int optional, requests objects of a certain type – items or containers. (OBJECT_UNKNOWN = -1,OBJECT_CONTAINER = 0,OBJECT_ITEM = 1), by default - OBJECT_UNKNOWN (e.g. all object types)*
<item_type/> *- int optional, requests items of a certain type (ITEM_UNKNOWN = -1, ITEM_RECORDED_TV = 0, ITEM_VIDEO = 1, ITEM_AUDIO = 2, ITEM_IMAGE = 3), by default - ITEM_UNKNOWN (e.g. all item types)*
<start_position/> *- int optional, by default - 0*
<requested_count/> *- int optional, by default "-1" (e.g. all)*
<children_request/> *- bool optional, by default – false (if false – returns information about object itself as specified by its object_id. If true – returns objects' children objects – containers and items)*
<server_address/> *- string mandatory (ip address or a host name of DVBLink server)*
</object_requester>

Response xml_result

<object xmlns="http://www.dvblogic.com">
<containers>
<container>
<object_id/> *- string mandatory*
<parent_id/> *- string mandatory*
<name/> *- string mandatory*
<description/> *- string optional*
<logo/> *- string optional*
<container_type/> *- int mandatory (CONTAINER_UNKNOWN = -1,CONTAINER_SOURCE = 0, CONTAINER_TYPE = 1, CONTAINER_CATEGORY = 2, CONTAINER_GROUP = 3)*
<content_type/> *- int mandatory, defines type of items in this container (ITEM_UNKNOWN = -1, ITEM_RECORDED_TV = 0, ITEM_VIDEO = 1, ITEM_AUDIO = 2, ITEM_IMAGE = 3)*
<total_count/> *- int optional*
<source_id/> *- string optional, identifies a physical source of this container (8F94B459-EFC0-4D91-9B29-EC3D72E92677 – is the built-in dvblink recorder, e.g. Recorded TV items)*
</container>
…
</containers>
<items>
<recorded_tv>
<object_id/> *- string mandatory*
<parent_id/> *- string mandatory*
<url/> *- string mandatory, url containing http link for stream playback*
<thumbnail/> *- string mandatory, url containing a hyperlink to item's thumbnail*
<can_be_deleted/> *- bool optional, identifies whether this item can be deleted*
<size/> *- long optional, item file size in bytes*
<creation_time/> *- long optional*
<channel_name/> *- string optional*
<channel_number/> *- int optional*
<channel_subnumber/>*- int optional*
<channel_id/>*- string mandatory*
<schedule_id/>*- string mandatory*
<schedule_name/>*- string mandatory*
<schedule_series/>*- bool mandatory, shows if this recorded tv item was part of series recording*
<state/> *- int optional (RTVS_IN_PROGRESS = 0, RTVS_ERROR = 1, RTVS_FORCED_TO_COMPLETION = 2, RTVS_COMPLETED = 3). State of the recorded TV item: being recorded, not even started because of error, recorded, but may miss certain part at the end because it was cancelled by user, completed successfully.*

<video_info> *- mandatory*
<name/> *— string mandatory*
<start_time/> *— long mandatory*
<duration/> *— int mandatory*

string optional:
<short_desc/>
<subname/>
<language/>
<actors/>
<directors/>
<writers/>
<producers/>
<guests/>
<categories/>
<image/>

int optional:
<year/>
<episode_num/>
<season_num/>
<stars_num/>

<starsmax_num/>

bool optional:
<hdtv/>
<premiere/>
<repeat/>
<cat_action/>
<cat_comedy/>
<cat_documentary/>
<cat_drama/>
<cat_educational/>
<cat_horror/>
<cat_kids/>
<cat_movie/>
<cat_music/>
<cat_news/>
<cat_reality/>
<cat_romance/>
<cat_scifi/>
<cat_serial/>
<cat_soap/>
<cat_special/>
<cat_sports/>
<cat_thriller/>
<cat_adult/>
</video_info>
</recorded_tv>

<video>
<object_id/> *- string mandatory*
<parent_id/> *- string mandatory*
<url/> *- string mandatory, url containing http link for stream playback*
<thumbnail/> *- string mandatory, url containing a hyperlink to item's thumbnail*
<can_be_deleted/> *- bool optional, identifies whether this item can be deleted*
<size/> *- long optional, item file size in bytes*
<creation_time/> *- long optional*

<video_info> *- mandatory, the same as for recorded_tv item*
…
</video_info>
</video>

<audio>
Currently not supported
</audio>

<image>
Currently not supported
</image>
…
</items>
<actual_count/> *- int optional, number of items and containers in this response*
<total_count/> *- int optional, total number of items and containers in the container*
</object>

Transcoded playback of the recorded TV item

If server platform supports transcoding then client can request a transcoded playback of the recorded TV item. In this case the recording is delivered by the server in HLS format.
To request transcoded playback the url has to be modified as following:
New url is <url> +
&client=**<client_id>**&transcoder=hls&height=**<height>**&width=**<width>**&bitrate=**<bitrate>**&lng=**<lng_code>**

where

- <url> - is the original url of the playback item as received from the server
- <client_id> - a unique id of the client that requests a stream. Can be any string of alphanumeric characters without spaces as long as it is unique – a guid for example
- <height> - Frame height of the transcoded stream
- <width> - Frame width of the transcoded stream
- <bitrate> - Bitrate in kbits/sec of the transcoded stream
- <lng> - (optional) an ISO 639 code of the language

For example the string to add to url can be
&client=AAABBBCCC&transcoder=hls&height=480&width=640&bitrate=512&lng=eng

Function remove_object

DVBLink command
remove_object

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<object_remover xmlns="http://www.dvblogic.com">
    <object_id/> - *string mandatory*
</object_remover>

Response xml_result
Response contains only status_code.

Function stop_recording

DVBLink command
stop_recording

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<stop_recording xmlns="http://www.dvblogic.com">
    <object_id/> *- string mandatory*
</stop_recording>

Response xml_result
Response contains only status_code.

Function get_streaming_capabilities

DVBLink command
get_streaming_capabilities

This function returns streams and protocols actually supported by a given instance of DVBLink server. Also general server streaming properties – like timeshift support and whether server can record – are returned.
The response contains supported protocols and transcoders for live TV and for playback items (e.g. TV recordings).

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<streaming_caps xmlns="http://www.dvblogic.com" />

Response xml_result
<streaming_caps xmlns="http://www.dvblogic.com">
    <protocols/> *- int (bitmask) mandatory (UNSUPPORTED_PROTOCOLS = 0, HTTP_TYPE = 1, UDP_TYPE = 2, RTSP_TYPE = 4, ASF_TYPE = 8, HLS_TYPE = 16, WEBM_TYPE = 32, MP4_TYPE = 64,*

ALL_SUPPORTED_PROTOCOLS = 65535)
<transcoders/> *- int (bitmask) mandatory (UNSUPPORTED_TRANSCODERS = 0, WMV_TYPE = 1, WMA_TYPE = 2, H264_TYPE = 4, AAC_TYPE = 8, RAW_TYPE = 16, VPX_TYPE = 32, VORBIS_TYPE = 64, ALL_SUPPORTED_TRANSCODERS = 65535)*
<pb_protocols/> *- int (bitmask) mandatory (UNSUPPORTED_PROTOCOLS = 0, HTTP_TYPE = 1, UDP_TYPE = 2, HLS_TYPE = 16, ALL_SUPPORTED_PROTOCOLS = 65535)*
<pb_transcoders/> *- int (bitmask) mandatory (UNSUPPORTED_TRANSCODERS = 0, H264_TYPE = 4, AAC_TYPE = 8, RAW_TYPE = 16, ALL_SUPPORTED_TRANSCODERS = 65535)*
<addressees> *- list of the registered DVBLink message addressees*
<addressee>
<id/> *- string mandatory, id of the DVBLink message addressee*
</addressee>
…
</addressees>
</can_record/> *- boolean optional. If present and its value is true then server can record content. Otherwise only live TV functionality is available.*
<supports_timeshift/> *- boolean optional. If present and its value is true then server supports timeshifting functionality*
<timeshift_version/> *- int optional. If present, provides of the timeshift algorithm implementation.*
</device_management/> *- boolean optional. If present and its value is true then server exposes device management part of the API (get_devices, get_scanners, channel_scan etc.).*
</streaming_caps>

Function get_recording_settings

DVBLink command
get_recording_settings

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<recording_settings xmlns="http://www.dvblogic.com" />

Response xml_result
<recording_settings xmlns="http://www.dvblogic.com">
<before_margin/> *- int mandatory, before margin in seconds*
<after_margin/> *- int mandatory, after margin in seconds*
<recording_path/> *- string mandatory*
<total_space/> *- long mandatory, in KB*
<avail_space/> *- long mandatory, in KB*
<check_deleted/> *- bool mandatory, indicates if server performs background check for deleted recording files*
<ds_auto_mode/> *- bool mandatory, indicates if DVBLink uses built-in algo to calculate minimum disk space size for warning and auto-delete algorithm*
<ds_man_value/> *- long mandatory, current minimum disk space value in KB*
<auto_delete/> *- bool mandatory, indicates if autodelete algorithm is enabled (delete oldest recordings when disk space size is less than threshold value)*
<new_only_algo_type/> *- int mandatory, indicates how “new only” schedule flag is treated by the recording engine. Values: 0 – as not recorded before, 1 – as not having “is_repeat” flag in program metadata*
</recording_settings>

Function set_recording_settings

DVBLink command
set_recording_settings

Request xml data
<recording_settings xmlns="http://www.dvblogic.com">
<before_margin/> *- int mandatory, in seconds*
<after_margin/> *- int mandatory, in seconds*
<recording_path/> *- string mandatory*
<check_deleted/> *- bool mandatory, indicates if server performs background check for deleted recording files*

<ds_auto_mode/> *- bool mandatory, indicates if DVBLink uses built-in algo to calculate minimum disk space size for warning and auto-delete algorithm*
<ds_man_value/> *- long optional, minimum disk space value in KB in case when ds_auto_mode is false*
<auto_delete/> *- bool mandatory, indicates if autodelete algorithm is enabled (delete oldest recordings when disk space size is less than threshold value)*
<new_only_algo_type/> *- int mandatory, indicates how "new only" schedule flag is treated by the recording engine. Values: 0 – as not recorded before, 1 – as not having "is_repeat" flag in program metadata*
</recording_settings>

Response xml_result
Response contains only status_code.

Function get_server_info

DVBLink command
get_server_info

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<server_info xmlns="http://www.dvblogic.com" />

Response xml_result
<server_info xmlns="http://www.dvblogic.com">
<install_id/> *- string mandatory, id of the DVBLink server installation*
<server_id/> *- string mandatory, id of the DVBLink Server product*
<version/> *- string mandatory, DVBLink Server version (format x.x.x)*
<build/> *- int mandatory, DVBLink Server build*
</server_info>

Function execute_command

This function allows sending custom commands to specific DVBLink modules. The actual list of modules, their supported commands and input/output parameters are listed below in a separate chapter.

DVBLink command
execute_command

Request xml data
< xml_cmd xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
< addressee/> *string mandatory – id (uuid) of the DVBLink module to send a command to*
< cmd/> *string mandatory – command*
<param/> *string optional, input parameters in xml format. The list, types and presence of input parameters depends on the command itself*
</xml_cmd>

Response xml_result
<xml_response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
<result/> *- string mandatory – command result. Two values – "success" and "fail" are predefined, but module can define additional custom values for each particular command.*
<param/> *- string optional – output parameters of the command in xml format. The list, types and presence of output parameters depends on the command itself*
</xml_response>

Function get_channel_url

DVBLink command
get_channel_url

Function returns a list of direct streaming urls for a set of the channel ids.

Request xml data
< stream_info xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<client_id/> *string optional - a unique id of the client that requests a stream. Can be any string of alphanumeric characters without spaces as long as it is unique – a guid for example. If empty or not present, client id will be generated from the client's IP address*
<channels_dvblink_ids>
<channel_dvblink_id/> *- long mandatory – DVBLink channel ID*
…
<channel_dvblink_id/>
</channels_dvblink_ids>
</ stream_info>

Response xml_result
< stream_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
<channel>
<channel_dvblink_id/> *- long mandatory – DVBLink channel ID*
<url/> *- string mandatory – direct streaming url*
</channel>
…
<channel>
</channel>
</ stream_info>

Function timeshift_get_stats

DVBLink command
timeshift_get_stats

Function returns statistic of currently running timeshifted streaming. This function only applies to indirect live streams, started with stream types "raw_http_timeshift" and "h264ts_http_timeshift".

Request xml data
< timeshift_status xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<channel_handle/> *- long mandatory, channel handle to obtain the timeshift stats for*
</ timeshift_status>

Response xml_result
< timeshift_status xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.dvblogic.com">
< max_buffer_length /> *- uint64 mandatory – maximum size of the timeshift buffer in bytes*
< buffer_length /> *- uint64 mandatory – current size of the timeshift buffer in bytes (may be less than max_buffer_length while buffer is growing)*
< cur_pos_bytes /> *- uint64 mandatory – current playback position within the timeshift buffer in bytes (range: between 0 and buffer_length)*
< buffer_duration /> *- uint64 mandatory – current duration of the timeshift buffer in seconds*
< cur_pos_sec /> *- uint64 mandatory – current playback position within the timeshift buffer in seconds (range: between 0 and buffer_duration)*
</ timeshift_status >

Function timeshift_seek

DVBLink command
timeshift_seek

Function positions playback pointer within timeshift buffer either by time or by bytes. This function only applies to indirect live streams, started with stream types “raw_http_timeshift” and “h264ts_http_timeshift”.

Request xml data
< timeshift_seek xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<channel_handle/> *- long mandatory, channel handle*
<type/> *- long mandatory, type of seek operation: 0 – by bytes, 1 – by time*
<offset/> *- int64 mandatory, offset in bytes (for seek by bytes) or in seconds (for seek by time). Offset may be negative value and is calculated from a position, given by whence parameter*
<whence/> *- long mandatory: 0 – offset is calculated from the beginning of the timeshift buffer, 1 – offset is calculated from the current playback position, 2 – offset is calculated from the end of the timeshift buffer*
</ timeshift_seek>

Response xml_result
Response contains only status_code.

Notes
Before issuing timeshift_seek command, it is recommended that client stops reading from the url, returned by play_channel command and resumes reading from a new position once timeshift_seek command completes.

Function get_devices (provisional)

DVBLink command
get_devices

This function returns devices, detected by DVBLink Server.

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<devices xmlns="http://www.dvblogic.com" />

Response xml_result
<devices xmlns="http://www.dvblogic.com">
<device>
<id>XXX-XXX-XXXXX</id>*- string, mandatory*
<name>DVBLogic TVButler</name>*- string, mandatory*
<state>new</state>*- string, mandatory. Possible values: new (present, no channels), active (present, with channels), missing (absent, with previously scanned channels), error*
<status>idle</ status>*- string, mandatory. Possible values: idle, streaming, scanning, networks_scanned, scan_finished*
<standards>0</standards>*- optional, contains bitmask value with supported broadcast standards: unknown (0x00000000), dvbt (0x00000001), dvbc (0x00000002), dvbs (0x00000004), atsc (0x00000008), cqam (0x00000010), iptv (0x00000020)*
<providers>*- list of providers that are available on this device*
<provider>
<provider_id>XXX-XXX-XXXXX</ provider_id>*- string, mandatory, provider id*
< provider_name>DVBLogic TVButler</ provider_name>*- string, mandatory, provider name*
</provider>
....
</providers>
</device>

```
        <device>
                ....
        </device>
</devices>
```

Function get_scanners (provisional)

DVBLink command
get_scanners

This function returns list of scan settings that are applicable for a specific device.
The return value enables generation of a parameterized user interface, which allows user selection of the scan settings for a given device.
Each list of possible settings applies for a particular broadcast standard that device supports. By default, the autodetected list of standards is used. It is possible to overwrite the autodetection by providing a specific standard as part of the request.
All settings have the same structure: setting has an id, human readable name and a list of possible values. Each value is also identified by its id and has a human readable name. Value may also contain a desc attribute with a human readable description of the value.
The selected values are passed to the start_scan function in form of <id>value</id> format.

Request xml data

```
< scanners xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <device>xxxx</device> - string, mandatory. Id of the device to get scanners for
        <standard>xxxx</standard > - string, optional. If present, overwrites the auto-detected device standards. Values: unknown (0x00000000), dvbt (0x00000001), dvbc (0x00000002), dvbs (0x00000004), atsc (0x00000008), cqam (0x00000010), iptv (0x00000020)
</scanners>
```

Response xml_result

```
<scanners xmlns="http://www.dvblogic.com">
        <standard id="1" name="DVB-T">- mandatory, both attributes are mandatory. Identifies the standard for which list of providers and scan settings are returned
                <settings>
                        <setting id="provider" name="Scan providers">
                                <value id="xxxx" name="Digitenne" desc=”Transponder scan”/>
                                <value id="xxxx" name="Freeview"/>
                                ....
                        </setting>
                </settings>
        </standard>
        <standard id="2" name="DVB-C">
                <settings>
                        <setting id="provider" name="Scan providers">
                                <value id="xxxx" name="Ziggo" />
                                <value id="xxxx" name="Caiway"/>
                                <container id="xxxx" name="Fast scan (manual entry)” desc=”Test container” >- container element presents an extra level of hierarchy with several user-editable elements inside and Ok button to confirm the user selection
                                        <edit id="xxxx" name="Frequency" format="number">12720</edit>- edit box. Format field can have values “number” or “string”. It has the value, displayed by default in the edit box.
                                        <edit id="xxxx" name="Symbol rate" format="number">6875</edit>
                                        <selection id="xxxx" name="Modulation" >- combobox, allowing selection of a single element out of the list
                                                <value id="xxxx" name="QAM64" />
                                                <value id="xxxx" name="QAM128" />
                                                <value id="xxxx" name="QAM256" />
                                        </selection>
                                </container>
                                ....
```

```
                </setting>
            </settings>
        </standard>
        <standard id="3" name="DVB-S/S2">
            <settings>
                <setting id="provider" name="Scan providers">
                    <value id="xxxx" name="Astra 19" />
                    <value id="xxxx" name="Astra 23"/>
                    ....
                </setting>
                <setting id="diseqc" name="Diseqc selection">
                    <value id="none", name="None"/>
                    <value id="simple_a", name="Simple A"/>
                    .....
                </setting>
                ....
            </settings>
        </standard>
        ....
</scanners>
```

Function start_scan (provisional)

DVBLink command
start_scan

This function starts channel scan on the given device. The scan parameters are defined by the user selection of the parameters, specified in get_scanners function call (provider key) or by get_networks function (network key).

Request xml data

```
<start_scan xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <device>xxxx</device> - string, mandatory. Id of the device.
        <provider>xxxx</provider> - mandatory. Id of the selected provider (exclusive with network)
        <network>xxxx</ network > - mandatory. Id of the selected network (exclusive with provider)
        ...
</start_scan>
```

Response xml_result
Response contains only status_code.

Function cancel_scan (provisional)

DVBLink command
cancel_scan

This function cancels ongoing channel scan on the given device.

Request xml data

```
<cancel_scan xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <device>xxxx</device> - string, mandatory. Id of the device.
</cancel_scan>
```

Response xml_result
Response contains only status_code.

Function apply_scan (provisional)

DVBLink command
apply_scan

This function applies scan results (e.g. saves scanned channels) on the given device.

Request xml data
<apply_scan xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<device>xxxx</device> *- string, mandatory. Id of the device.*
</apply_scan>

Response xml_result
Response contains only status_code.

Function get_networks (provisional)

DVBLink command
get_networks

This function returns list of scanned networks after inital scanning of networks is finished (device has in networks_scanned status). The selected value has to be passed to a start_scan function.

Request xml data
<networks xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com">
<device>xxxx</device> *- string, mandatory. Id of the device to get scanners for*
</networks>

Response xml_result
<networks xmlns="http://www.dvblogic.com">
<settings>
<settings id="network" name="Scanned networks">
<value id="40350" name="Ziggo, area A" />
<value id="40351" name="Ziggo, area B"/>
....
</setting>
</settings>
</networks>

Function get_device_status (provisional)

DVBLink command
get_device_status

This function returns detailed status of a given device

Request xml data
< device_status xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<device>xxxx</device> *- string, mandatory. Id of the device*
</device_status>

Response xml_result
<device_status xmlns="http://www.dvblogic.com">
<status>idle</ status>*- string, mandatory. Possible values: idle, streaming, scanning, networks_scanned, scan_finished*
<scanning>*- this part is only applicable of device is in scanning, networks_scanned and scan_finished states*
<progress>10</progress>*- long, mandatory. Percentage of the completed scan*
<channels>198</channels>*- long, mandatory. Number of found channels*

```
            <transponders>- list of scanned transponders
                <transponder id="xxx" lock="1" strength="10" quality="100">NPO1, NPO2,
NPO3</transponder>- id: string,mandatory, lock: long, mandatory (values 0, 1), strength: long, mandatory
(signal strength 0-100), quality: long, mandatory (signal quality 0-100). Value - human readable scan status,
usually list of found channels
                <transponder id="xxx" lock="0" strength="0" quality="0"> </transponder>
            <transponders>
        </scanning>
        <streaming>- this part is applicable is device is streaming
            <channel id="xxx" lock="1" strength="10" quality="78"/>- information about channel, being
streamed. id: string, mandatory (channel id), lock: long, mandatory (values 0, 1), strength: long, mandatory
(signal strength 0-100), quality: long, mandatory (signal quality 0-100)
            ...
        </streaming>
</device_status>
```

Function repair_database (provisional)

DVBLink command
repair_database

This function repairs recording database on a server.

Request xml data

```
<repair_database xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
</repair_database>
```

Response xml_result
Response contains only status_code.

Function force_epg_update (provisional)

DVBLink command
force_epg_update

This function forces epg update (rescan or re-fetch from the internet source).

Request xml data

```
<force_epg_update xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
</force_epg_update>
```

Response xml_result
Response contains only status_code.

Function get_channels_visibility (provisional)

DVBLink command
get_channels_visibility

This function returns list of invisible channels, each identified by its id.

Request xml data

```
<invisible_channels xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
</invisible_channels>
```

Response xml_result
<invisible_channels xmlns="http://www.dvblogic.com">
<channel id="4:546000:1:5:5501" />
<channel id="4:546000:1:5:5502" />
....
</invisible_channels>

Function set_channels_visibility (provisional)

DVBLink command
set_channels_visibility

This function sets the list of invisible channels, each identified by its id.

Request xml data
<invisible_channels xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<channel id="4:546000:1:5:5501" />
<channel id="4:546000:1:5:5502" />
....
</invisible_channels>

Response xml_result
Response contains only status_code.

Function get_epg_sources (provisional)

DVBLink command
get_epg_sources

This function returns list of the available sources of EPG information.

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<epg_sources xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com" />

Response xml_result
<epg_sourcesxmlns="http://www.dvblogic.com">
<source id="<source id>" name ="<source name>" settings="true" /> *- each source row has mandatory id and name attributes of type string. “settings” attribute is optional and, if present, indicates that source has configurable settings*
....
</epg_sources>

Function get_epg_channels (provisional)

DVBLink command
get_epg_channels

This function returns list of the available channels with epg information for a given epg source.

Request xml data
<epg_channels xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<epg_source_id>xxxx</epg_source_id> *- string, mandatory. Id of the epg source to get the channels for*
</epg_channels>

Response xml_result

```
<epg_channels xmlns="http://www.dvblogic.com">
        <channel id="xxx" name="xxx" num="x" subnum="x" > - each channel row has mandatory attributes id and name of type string. Attributes num and subnum are optional and have type long
                <logo>http://x.x.x.x/aaa</logo> - logo is optional. The value itself is uri string
        </channel >
                ....
</epg_channels>
```

Function get_epg_channel_config (provisional)

DVBLink command

get_epg_channel_config

This function returns epg channel configuration for a list of the requested channels.

Request xml data

```
<epg_channels_config xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <channel id="4:546000:1:5:5501" />  - each channel is identified by its mandatory id string attribute
        <channel id="4:546000:1:5:5502" />
        ....
</ epg_channels_config>
```

Response xml_result

```
<epg_channels_config xmlns="http://www.dvblogic.com">
        <channel id="xxx" > - each channel is identified by its mandatory id string attribute
                <epg_source_id>xxxxx</epg_source_id> - epg source id. String, mandatory
                <epg_channel_id>xxxxx</epg_channel_id> - epg channel id. String, mandatory
        </channel >
                ....
</epg_channels_config>
```

Note: By default each channel gets its EPG information from its own device (e.g. from the stream). The default epg source id is **e96f83f5-79ad-4ea4-af2b-6682f233ad1a**. When default epg source id is used, epg_channel_id is set to the actual id of this channel.

Function set_epg_channel_config (provisional)

DVBLink command

set_epg_channel_config

This function sets epg channel configuration.

Request xml data

```
<epg_channels_config xmlns="http://www.dvblogic.com">
        <channel id="xxx" > - each channel is identified by its mandatory id string attribute
                <epg_source_id>xxxxx</epg_source_id> - epg source id. String, mandatory
                <epg_channel_id>xxxxx</epg_channel_id> - epg channel id. String, mandatory
        </channel >
                ....
</epg_channels_config>
```

Response xml_result

Response contains only status_code.

Note: To reset epg for a particular channel to default the epg_source_id for this channel has to be set to

e96f83f5-79ad-4ea4-af2b-6682f233ad1a and epg_channel_id has to be set to the actual id of this channel. It is also allowed just to skip this channel in the request data.

Function match_epg_channels (provisional)

DVBLink command
match_epg_channels

This finds a matching EPG channel(s) inside a particular epg source for a list of the requested channels. For matching purpose, function uses a number of internal parameters, such as nid-tid-sid, channel name etc. The results are returned as a list of exactly matched and/or partially matched channels.

Request xml data

```
<match_info xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.dvblogic.com">
        <epg_source_id>xxxxx</epg_source_id> - epg source id to use for matching. String, mandatory. If default epg source id is specified, matching is done against all available epg sources
        <channels>
                <channel id="4:546000:1:5:5501" />  - each channel is identified by its mandatory id string attribute
                <channel id="4:546000:1:5:5502" />
                ....
        </channels>
</match_info>
```

Response xml_result

```
<match_info xmlns="http://www.dvblogic.com">
        <channel id="xxx" > - each channel is identified by its mandatory id string attribute
                <exact>- epg channel, exactly matching the requested channel
                        <epg_channel epg_source_id ="xxx" epg_channel_id ="xxx" />
                </exact>
                <partial> epg channels, partially matching the requested channel
                        <epg_channel epg_source_id ="xxx" epg_channel_id ="xxx" />
                        ....
                </partial>
        </channel >
                ....
</match_info >
```

Function get_installed_products (provisional)

DVBLink command
get_installed_products

This function returns list of the installed products and information about them.

Request xml data

```
<?xml version="1.0" encoding="utf-8" ?>
<installed_products xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com" />
```

Response xml_result

```
<installed_products xmlns="http://www.dvblogic.com">
        <product id="xxx" > - each product row has mandatory id attribute of type string.
                <name>DVBLink TVSource</name> - product name (string), mandatory
                <version>6.0.0</version> - product version (string), mandatory
                <build>15123</build> - product build (string), mandatory
                </trial_available> - optional, if present indicates that product has trial functionality
                </requires_registration> - optional, if present indicates that product requires registration
```

</requires_subscription> *- optional, if present indicates that product requires subscription*
</requires_coupon> *- optional, if present indicates that product has to be activated using a coupon code*
</activation_in_progress> *- optional, if present indicates that this product is being activated (either product license, subscription or trial)*
<days_left>12</ days_left > *- optional, for products with subscription or active trial indicates number of days left (int)*
<license_state>free</ license_state > *- mandatory,string, indicates license state of the product. Possible values are: "wrong_fp", "free", "trial", "registered", "expired", "no_license_file", "no_subscription", "subscribed", "subscription_expired", "subscription_wrong_fp", "no_coupon", "coupon_wrong_fp"*
<license_name>xxxx.lic</license_name> *- optional, string, indicates name of the license file*
<license_key>0x123456</license_key> *- optional, string, indicates license id*
<fingerprint>AAAA-BBBB</fingerprint > *mandatory, string, indicates machine fingerprint*
<machine_id>1123</machine_id> *optional, string, if present indicates machine id for which license was generated*
</product>
....
</installed_products >

Function activate_product (provisional)

DVBLink command
activate_product

This function requests product activation.

Request xml data
<activate_product xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<id>1234-3456-xxx</id> *- mandatory, string, id of the product to be activated*
<serial>AAAA-BBBB-CCCC</ serial > *- mandatory, string, serial number (aka coupon) to activate the product*
<email>a@a.com</email> *- mandatory, string, e-mail address of the user, activating product*
<user_name>a@a.com</user_name> *- optional, string, name of the user, activating product*
</activate_product >

Response xml_result
<activate_product xmlns="http://www.dvblogic.com">
<result>success</result>*- mandatory, string, result of the activation operation. Possible values are: "success", "no_server_connection", "no_activations_available", "already_activated", "invalid_login", "in_progress", "file_write_error", "error", "invalid_xml", "invalid_data", "invalid_coupon", "already_used_coupon","email_already_in_use", "other_product_coupon"*
</activate_product >

Notes:if function returns “in_progress” as a result then server will complete the activation request in the background. Status of the operation can be monitored using “get_installed_products” command and its “activation_in_progress” member.

Function activate_product_trial (provisional)

DVBLink command
activate_product_trial

This function requests product trial activation.

Request xml data
<activate_product_trial xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com">
<id>1234-3456-xxx</id> *- mandatory, string, id of the product to be activated*
</activate_product_trial>

Response xml_result
<activate_product xmlns="http://www.dvblogic.com">
<result>success</result>*- mandatory, string, result of the activation operation. Possible values are: "success", "no_server_connection", "no_activations_available", "already_activated", "invalid_login", "in_progress", "file_write_error", "error", "invalid_xml", "invalid_data", "invalid_coupon", "already_used_coupon","email_already_in_use", "other_product_coupon"*
</activate_product >

Notes:if function returns “in_progress” as a result then server will complete the activation request in the background. Status of the operation can be monitored using “get_installed_products” command and its “activation_in_progress” member.

Function updater_get_status (provisional)

DVBLink command
updater_get_status

This function returns list of the installed products and information about them.

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<updater_status xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com" />

Response xml_result
< updater_status xmlns="http://www.dvblogic.com">
<status>up_to_date</status> *- mandatory, string, indicates updater status. Possible values are: "up_to_date", "needs_update", "update_in_progress"*
<components> *- lists components, for which updates are checked by server*
<component>
<name>resources</name> *- mandatory, string, component name*
<local>11000</ local > *- mandatory, int, available local revision. If 0 – revision is unknown*
< remote >12200</remote> *- mandatory, int, available remote revision. If 0 – revision is unknown*
<changes>…..</changes> *- optional, string, description of changes*
</component>
……
</components>
</updater_status>

Function updater_check_update (provisional)

DVBLink command
updater_check_update

This function checks if updates are available. Results can be retrieved using “updater_get_status” command.

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<updater_check_update xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.dvblogic.com" />

Response xml_result
Response contains only status_code.

Function updater_start_update (provisional)

DVBLink command
updater_start_update

This function performs actual update.

Request xml data
<?xml version="1.0" encoding="utf-8" ?>
<updater_check_update xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.dvblogic.com" />

Response xml_result
Response contains only status_code.

DVBLink modules and their commands

Social module

Module ID: a8b42f8e-6a08-418a-8917-e767f766d576

Get_targets command

Returns list of SendTo targets

Command
get_targets

Input parameters
None

Output parameters

<get_targets>
    <target>
        <id/> *- string mandatory – target id*
        <name/> *- string mandatory – target name*
        <delete_on_success/> *- bool mandatory – indicates if original playback item should be deleted on success*
        <frm_id/> *- string mandatory – formatter id*
        <frm_params/> *- string mandatory – formatter parameters*
        <dst_id/> *- string mandatory – destination id*
        <dst_params/> *- string mandatory – destination parameters*
    </target>
    <target>
        …
    </target>
</get_targets>

Send_to_add_item command

Adds playback item to the SendTo processing queue

Command

send_to_add_item

Input parameters
<send_to_add_item>
	<item>
		<object_id/> *- string mandatory – playback item object id*
		< description /> *- string optional – description for this item. If empty, it will be generated automatically*
		< target /> *- string mandatory – SendTo target id*
	</item>
	<item>
		…
	</item>
</send_to_add_item>

Output parameters
<send_to_add_item>
	<item_id/> *- string mandatory – id of the newly added item*
	…
	<item_id/>
</send_to_add_item>

Send_to_get_items command

Returns list of items from the SendTo processing queue

Command
send_to_get_items

Input parameters
<send_to_get_items>
	<type/> *- int mandatory – type of the items to return: 0 – all items, 1 – active items, 2 – completed items*
</send_to_get_items>

Output parameters
<send_to_get_items>
	<item>
		<item_id/> *- string mandatory –item id*
		<object_id/> *- string mandatory – playback item object id*
		< description /> *- string optional – description for this item*
		<created/> **–** *mandatory long – timestamp when this item was created*
		< target /> *- string mandatory – SendTo target id*
		<completed/> **–** *mandatory long – timestamp when this item was completed. If not completed yet, this field equals to 0*
		< status /> *- int mandatory – item status: 0 – pending, 1 – fetching, 2 – formatting, 3 – sending, 4 – completed success, 5 – completed fail, 6 - cancelled*
	</item>
	<item>
		…
	</item>
</send_to_get_items>

Send_to_cancel_item command

Cancels active SendTo item (completes it with a cancelled state)

Command
send_to_cancel_item

Input parameters
<send_to_cancel_item>
<item_id/> *- string mandatory – id of the item to cancel*
</send_to_cancel_item>

Output parameters
None

Multicast streamer module

Module ID: e4a0f9ae-79d3-4512-a1d6-01aba7435e98

Get_mcast_url_cmd command

Return multicast url for a set of channel ids

Command
get_mcast_url_cmd

Input parameters
<get_mcast_url_cmd>
<channel/> *- long mandatory – DVBLink channel id*
…
<channel/>
</get_mcast_url_cmd>

Output parameters
<get_mcast_url_cmd>
< channel >
<id/> *- long mandatory – channel id*
<url/> *- string mandatory – channel multicast url*
</ channel >
<channel>
…
</ channel >
</get_mcast_url_cmd>