# LOEWE TV
# remote API

Author(s):

Timo Weggen
Oliver Kutter
Thorsten Habelitz

Timo Weggen

Tel.: (049)-511-563549-46
mailto: timo.weggen@loewe.de

Thorsten Habelitz

Tel: (049)-6151-39694-57
mailto: thorsten.habelitz@loewe.de

# 1 Revision History

| Version | Date | Editor | Department | Comment |
|---|---|---|---|---|
| 0.0.9 | 30.01.11 | TW | Middleware | Added player |
| 0.0.12 | 13.04.11 | TW | Middleware | Added udp/subscription related definitions. |
| 0.0.14 | 26.04.11 | TW | Middleware | EPG / search relations. |
| 0.0.16 | 27.04.11 | TW | Middleware | Cleaned up a bit to release preliminary information. |
| 0.0.19 | 12.05.11 | TW | Middleware | Renamed GetItemInfo in GetMediaItem, following the wsdl |
| 0.0.22 | 05.08.11 | OK | Applications | Reworked Get/SetVolume, Added Injecting (15eyboard) Keys. |
| 0.0.23 | 22.08.11 | OK | Applications | Cleaned up a bit. |
| 0.0.24 | 07.09.11 | OK | Applications | Added section on programming timers. |
| 0.0.25 | 13.09.11 | OK | Applications | Adjusted section on injecting RC keys. |
| 0.0.26 | 24.09.11 | OK | Applications | Added GetDeviceData. |
| 0.0.27 | 04.10.11 | OK | Applications | Some cleanup, added implementation status, extended RequestAccess message. |
| 0.0.28 | 06.10.11 | OK | Applications | Added Result field in ZapToApplication and ZapToMedia |
| 0.0.29 | 10.10.11 | OK | Applications | Added GetCurrentPlayback method. Added mode “delayed” for injecting RC keys. |
| 1.0.30 | 10.02.12 | OK | Applications | Added note on “favlist” access via GetChannelList. Also added description of media item UUID format (see Implementation Note to 8.7.1) |
| 1.0.31 | 16.02.12 | OK | Applications | Added “ChannelListName” property to GetChannelList replies. |
| 1.0.32 | 24.02.12 | OK | Applications | Added Chapter on bridging between SOAP and JavaScript |
| 1.0.33 | 05.06.12 | OK | Applications | Clarified UUID Format for SL2xx and above. |
| 1.0.34 | 03.08.12 | OK | Applications | Added GetMute / SetMute methods. |
| 1.0.35 | 30.11.12 | OK | Applications | Added MAC address to GetDeviceData reply. |
| 1.0.36 | 30.01.13 | OK | Applications | Added Wake On LAN. |
| 1.0.37 | 24.07.13 | TH | Applications | Added LAN and WLAN MAC address to GetDeviceData reply. |
| 1.0.39 | 31.07.14 | TH | Applications | Added TV set location to method GetDeviceData reply. Added method SetActionField. Added table of supported key injection values. Added method GetListOfChannelLists. Modified GetDeviceData default values. Added GetFeature method. Added attributes to method GetMediaItem and cleaned up documentation. |
| 1.0.40 | 23.01.15 | TH | Applications | Added functions to manage personal channel lists. Added AV list to response of GetListOfChannelLists. Added description of GetMediaItem field Attributes. Added hash values to GetListOfChannelLists and GetChannelList. |
| 1.0.41 | 27.04.15 | TH | Applications | Added GetDRPlusArchive, ExportServiceLists, ImportServiceLists, PlayMultiroom, GetListOfVirtualLists, AddFavoriteListFromVirtual. Modified GetListOfChannelLists, GetDeviceData, GetFeature and GetMediaItem. |
| 1.0.42 | 16.09.15 | TH | Applications | Modified GetListOfChannelLists, GetListOfVirtualLists, GetChannelList, AddFavoriteListFromVirtual. Modified GetDeviceData. Added calls SetSetting and GetSettings. |

## 2 Related Documents

### 2.1 Normative references

| Document | Contents |
|---|---|
| http://tools.ietf.org/html/rfc2141 | URN definition |
| http://www.w3.org/TR/SOAP/ | SOAP definition |
| http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/ | WS addressing core specification |
| http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/ | WS addressing SOAP binding |
| http://specs.xmlsoap.org/ws/2004/09/soap-over-udp/soap-over-udp.pdf | SOAP over UDP |
| http://tools.ietf.org/rfc/bcp/bcp47.txt | IETF Language Codes |
| @@@ search me @@@ | UPnP directory service specification. |
| @@@ search me @@@ | id3v2 information |

### 2.2 Informative References

| Document | Contents |
|---|---|
| | |

### 2.3 Related LOEWE specifications

| Document | Contents |
|---|---|
| | |
| | |

## 3 Open Issues

## 4 Table of Contents

# 5 Introduction

## 5.1 Abstract

A single LOEWE TV is only one device within a larger home network. To enable interaction with other devices beyond the limits of standard protocols, the TV offers access to several specific functions via standard network.

To allow both interaction with unknown devices and to ensure secure operation with LOEWE devices, two channels of communication are defined: An open, mostly read-only channel, and a secure control channel.

This API specification is not restricted to the LOEWE slxxx series of TVs. It applies as well to TVs after this generation.

**Implementation Note**

It is important to note that a TV set is a very slow device. Mobile phones or other mobile devices are equipped with platforms that are magnitudes more powerful. When writing applications, try to cache whereever possible. Expect long latency times from the TV.

## 5.2 Portability and requirements

To ensure compatibility industry standard protocols are used whereever possible, e.g. the open channel uses SOAP over TCP, the secure channel uses SOAP over ssl.

# 6 Protocol

## 6.1 Transport level

The LOEWE TV listens on TCP port 905 for incoming http requests containing SOAP queries. SOAP over SMTP is not supported. Only `http/1.0` is mandatory, `http/1.1` is optional. Https support still is to be defined.

Throughout the documentation we assume the convention

```
xmlns:ltv="urn:loewe.de:RemoteTV:Tablet"
```

to make reading easier.

The LOEWE TV uses udp to send out update notifications. Updates are sent to a port individually chosen by each of the subscribers.

## 6.2 SOAP content

The messages are standard SOAP messages. The ws addressing and ws eventing standards are adopted to implement subscriptions, notifications and alike.

In all messages that use ws addressing, we assume the convention

```
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
```

to shorten the examples. Likewise, for ws eventing, we assume

```
xmlns:wse="http://www.w3.org/2009/02/ws-evt"
```

to omit the repeating clauses from every example.

## 6.3 Reused from UPnP

This specification reuses some UPnP definitions, like the media item type or the browse/search definitions.

# 7 Concepts

This chapter presents unsorted concepts used in this specification.

## 7.1 SOAP specifics

### 7.1.1 Subscription manager

This term originates in ws eventing lingo. It describes an address that can be used to perform a standard ws eventing subscription. Many objects carry a ws address to identify the subscription manager. When subscribing an object, a list of properties to monitor also must be passed to the object.

### 7.1.2 Events

@@@ write me, derived from WS eventing spec. @@@

### 7.1.3 SOAP over UDP

@@@ write me, referring the spec quoted in the doc header @@@

## 7.2 Media handling

This is a short practical overview, for a more in-depth discussion of media data types, refer to 10.1 : Media lifecycle.

### 7.2.1 Media items

Media items are things that represent a playable media, like an AVI file on a usb stick, a TV channel that can be watched, a track on a CD that can be played,l a track on an ipod that is connected, a youtube video etc. .

Media items are the basic things that given to players. Media items have a type, a location and a unique id.

Players can do a lot of things with media items, they cannot only display them, but can also extract other non-streaming information from media items, up to interactive applications.

### 7.2.2 Media information

Media information objects carry descriptive information about something that can be seen/heard, like title, long information, the theme and genre etc. . This is what is stored inside an mp3 file, what is broadcast about a TV show in EPG SI and what gracenote claims about a physical CD. Please note, that media information can vary during playback: On TV, some show is followed by another show, an internet radio station can change the current metainformation. Only some media have static media information, like an mp3 file, or a JPG image.

### 7.2.3 Media event

A media event tells you that a media information is available during a certain period of time. A media event is the abstract concept that is the same for an EPG event, a playback that is running etc. .

Media events are e.g. created by:

- A DVB EPG module that subscribes a player path's SI information and generates media event objects, storing them in a database, which eventually reaches the user as media events.
- An internet radio player (shoutcast etc.) that extracts metainfos from the stream, returning to media event subscriber.
- A proprietary NVoD Service that parses its stream availability infos
- A internet service based EPG query module.

## 7.3 TV operating modes

From a user perspective, the TV set resembles a device that can display one application at a time. These applications include:

- Watching TV channels (including channel specific applications like Freeview, HbbTV etc.)
- Selecting a TV channel
- Browsing EPG information
- Browsing Teletext pages
- Browsing Web Pages
- Listening to a radio station
- Browsing media directories
- Playing internet videos

This API allows an external device to request jump to a certain operation mode the same way any use does. The TV might very well reject this request if it is not able to perform the operation.

@@@ explain more in detail the "Bedienmodus" semantics @@@

## 7.4 Internationalization

A lot of textual information is subject to internationalization. The API allows to associate textual information with a language code. User interfaces may select to display a certain text.

In many xml descriptions, strings may be associated with a comma-separated list of IETF language codes. Omitting the language code specification means that no language information for the string is available.

## 7.5 List queries

This API contains several calls to query a set of results, like channel lists, EPG informations, volume listings etc. . To improve device latency time and to make client side caching easier, list queries always return *reference objects*. These objects contain a set of UUIDs that allow retrieving the actual data. In effect, this allows mapping on faster transport protocols in many cases. Also, it offers a meaningful ID to the client to implement efficient asynchronous on-demand loading techniques.

# 8 Types

This chapter defines the types used in the communication. While reading first, you might want to skip to chapter 9 .

One of the most basic types in this API is the Notification type. A notification describes something that happens somewhere. To select the notifications to receive, a communication partner can subscribe it, thus becoming a subscriber.

## 8.1 Subscriber

## 8.2 Notification

Notifications use the syntax as specified in the ws eventing standard.

## 8.3 Input events

The TV deals with interaction brought to the device by input events. These input events can be unbound to any particular target (@@@), bound to a specific target by its semantic nature ("Power off"), bound to a specific target by the context as seen by the TV ("Volume plus"), or bound to a specific target as requested by client ("Stop playback in main player").

@@@ specify generic input event syntax @@@

The input events described later in this document are special cases of generic input events. The syntax given there is application specific. Please do not expect this specific syntax to be supported in later, new commands.

## 8.4 Input event targets

Input events can have targets (see explanations at @@@ref Input Events). Targets can be both specified by using an id as defined by the TV, or by using some few well-defined target ids.

### 8.4.1 Legacy RC key events

```
<ltv:RCKeyEvent alphabet="l2700" mode="press" value="1"/>
```

### 8.4.2 Legacy special purpose input events

```
<ltv:ModifyLocalVolume mode="relative">2</ltv:ModifyLocalVolume>
```

…

```
<ltv:MuteLocal mode="relative">1</ltv:MuteLocal>
```

### 8.4.3 Unbound keyboard input

Unbound keyboard input are input events that are not targeted to a particular focussable interactive element. Instead, these input events are injected before any part of the input management assigns them to any input object, just like a real keyboard.

However, in contrast to a real keyboard, these events are not subject to national mapping.

### 8.4.4 Targeted keyboard input

@@@ not defined yet @@@

## 8.5 Media directory

A media directory is a service that implements listing and searching media items and media events. Media directories are represented by short, 4 letter alphanumeric identifier. Current media directories are:

| | |
|---|---|
| FSL2 | [technically] local file system devices. |
| DLN1 | A UPnP content directory server as implemented by one device. |
| UTA1 | A directory service listing videos on a site compatible with youtube.com . |
| VTUN | Lists internet radio stations and podcasts as offered by the vtuner online service. |
| AUP0 | Lists streaming audio and video as represented by the AUPEO online service. |
| FAV0 | Provides local user favorites. |
| CHL0 | Searching and listing TV channels. |

### 8.5.1 Media directory purposes

Several purposes are defined for volumes (every media directories internally is represented by a volume). Please note, that only volumes marked as PURPOSE_MEDIA_SOURCE and at least one of PURPOSE_BROWSABLE or PURPOSE_SEARCHABLE can act as media directories.

| | |
|---|---|
| PURPOSE_SYSTEM | This is a volume used internally in the system. |
| PURPOSE_SEALED_BOX | This volume is an integral part of a sealed box product. |
| PURPOSE_EXTERNAL | This volume is externally attached to a product. This means, communication with this volume could be captured and modified by an experienced engineer. |
| PURPOSE_GENUINE | This volume has been verified to be a genuine manufacturer's product. |
| PURPOSE_VIA_NETWORK | This volume referes to something connected via (IP based) network. |
| PURPOSE_VIA_LOGICAL_FS | This volume referes to something connected via logical file system (fsal2) |
| PURPOSE_UPDATE_SOURCE | This volume can be used as a source for system updates. |
| PURPOSE_PVR_ARCHIVE | This volume can be used as storage memory for PVR archive storage. |
| PURPOSE_PVR_TIMESHIFT | This volume can be used as a temporary storage memory for timeshifting. |
| PURPOSE_MEDIA_SOURCE | This volume can be used as a media source. |
| PURPOSE_MEDIA_STORAGE | This volume can be used as a storage for media. In other words, media files can be copied to and stored onto this volume. |
| PURPOSE_IMAGES | This volume is especially suited for the storage of picture files. |
| PURPOSE_MUSIC | This volume is especially suited for the storage of music files. |
| PURPOSE_VIDEOS | This volume is especially suited for the storage of videos. |
| PURPOSE_BROWSABLE | This volume accepts browse queries (i.e. select by ancestor). |
| PURPOSE_SEARCHABLE | This volume accepts search queries. |
| PURPOSE_EXTERNALLY_LAYOUTED | This volume is externally layouted. |

@@@ extend this @@@

The purpose of a given media directory is a combination of any of these purposes.

**Implementation Note**

Media directories are listed by the reggie service. The directory services internally are offered by the quern module.

## 8.6 Media creator

There is no media creator type. The media creator is a role in the design that technically is not referred to in any parts of the specification.

## 8.7 Media item

Media item are the basic data entity that can be played (or watched, or listened to).

Every media item is associated with a locator. This locator can be used to identify this item, e.g. to trigger a playback. The locator basically looks like a url (and in fact can be one).

Also, every media item has a caption, that is a short name.

```
<ltv:MediaItem itemClass="object.item.videoItem.channel.DVB.Cable">
  <ltv:uuid>chl0://lo-00-8d-1c-4a-35-69-0146-8976246863667</ltv:uuid>
  <ltv:Locator>chl0://localhost/live/8976246863667</ltv:Locator>
  <ltv:ShortCaption>ARD</ltv:ShortCaption>
  <ltv:Caption>Das Erste</ltv:Caption>
</ltv:MediaInformation>
```

describes a channel list entry.

Find below a table displaying the possible contents of media information items.

**Properties**

| | |
|---|---|
| Locator | A URL to find the media item. |
| uuid | A UUID to identify the media item within the scope of the media directory service. |
| Caption | A reasonably short name or title of the media suitable for display in a list. Recommended length is between 10 and 40 characters. |
| ShortCaption | A very short name or title suitable for a very packed display. Recommended length is below 10 characters. [optional] |
| ThunbnailURL | A URL to a thumbnail for this item. [optional] |

**Attributes**

| | |
|---|---|
| itemClass | The class of this media item. |

**Implementation Note**

Please note that there is no direct relationship between the internal ma::ItemInfo class family and the media item class. This relationship still has to be defined.

### 8.7.1 Media Item uuid

The media item id is composed of:

- an ID of the directory service that referenced the item id. Technically, these Ids are the internal volume source types as defined by the volume manager.
  Examples
  - "DLN1" : References the DLNA sources
  - "CHL0" : References the channel list query service
  - "FSL2" : References local media.
- A uuid string unique within the scope of the respective directory service.

The resulting string is unique with respect to the target device.

The media item uuid is not subject to internationalization.

**Implementation Note**

For the channel list query service on SL1xx and SL2xx, the UUIDs have the following format:

chl0:00-GGGGOOOOffffTTTTSSSS

with

- GGGG = For **SL1xx**: gcn in hexadecimal notation, **SL2xx and later** versions: frontend information.
- OOOO = onid in hex notation. **For analog channels contains the CNI instead.**
- TTTT = tsid in hex notation. **Always 0000 for analog channels.**
- SSSS = sid in hex notation. **Always 0000 for analog channels.**

Example:

chl0:00-00000001ffffffff6dca

yields gcn 0 (on SL1xx), onid 1, tsid 65535 and sid 28106.

## 8.8 Media information

Media information is a base class to provide meta information describing some content distributed at some given time in a media item. The specification tries not to overlap media information properties with the basic media item properties

```
<ltv:MediaInformation>
  <ltv:ShortInfo>The River</ltv:Caption>
  <ltv:ExtendedInfo ltv:lang="en">
    A journey into the depth of the Amazonas, exploring ...
  </ltv:ExtendedInfo>
  <ltv:ExtendedInfo ltv:lang="de">
    Eine Reise in die Welten des Amazonas. Entdecken Sie ...
  </ltv:ExtendedInfo>
</ltv:MediaInformation>
```

Find below a table displaying the possible contents of media information items.

| | |
|---|---|
| ShortInfo | A short description of the media. Recommended length is below 100 characters. [INT] |
| ExtendedInfo | An in-depth description of the media.<br>Recommended length is between 100 and 4000 characters. [INT] |
| Playtime | The length of this media in seconds, if available. |
| Genre | The topic of this event. |

| | |
|---|---|
| | Used for theme information as well as for id3v2 genre informations. Colon @@@or comma; vfy with DLNA@@@-separated list of genres. |

Media information items can carry more informational items.

@@@ add fields also found as search fields. @@@

@@@ define sub classes. @@@

## 8.9 Media provider

There is no media provider type. The media provider is a role in the design that technically is not referred to in any parts of the specification.

## 8.10 Media event

Media event is a base class.

It contains a media information object. It is associated with a media item.

A media event is a set of media information valid for a certain time on a given media item.

```
<ltv:MediaEvent>
  <ltv:MediaInformation>
    <ltv:ShortInfo>The River</ltv:ShortInfo>
    <ltv:ExtendedInfo ltv:lang="en">
      A journey into the depth of the Amazonas, exploring ...
    </ltv:ExtendedInfo>
    <ltv:ExtendedInfo ltv:lang="de">
      Eine Reise in die Welten des Amazonas. Entdecken Sie ...
    </ltv:ExtendedInfo>
  </ltv:MediaInformation>
  <ltv:Availability>
    <ltv:ScheduledTime startTime="1235142089" duration="3600"/>
  </ltv:Availability>
</ltv:MediaEvent>
```

Description:

| | |
|---|---|
| Information | One single media information entry. |
| Availability | The availability information. Standard availability event.<br>When returned in calls, endTime also is (artificially) provided. |

## 8.11 Event sources

Events can be generated by event sources. Typical event sources include a EPG directory, a program list, a PVR archive. → event browser in ui

Several calls query a set of media items. These calls always return part of a result only. To ensure consistency among the results to the caller, each of the views returned is associated with a sequence number. Result views with the same sequence number are part of the exactly same result set. Also, to allow caching and to speed up communication, the result views only return keys to the items, they never return the entire media items.

## 8.12 Result item list fragment

```
<ltv:ResultItemFragment sequenceNumber="9869569843634-cghdfgh-27346"
    totalResults="100" returnedResults="2" startIndex="52">
        <ltv:ResultItemReference uuid="chl0://lo-00-8d-1c-4a-35-69-0146-8976246863667"/>
        <ltv:ResultItemReference uuid="chl0://lo-00-8d-1c-3a-25-23-1245-1564790863345"/>
</ltv:ResultItemFragment>
```

The above result item fragment contains two result items. To query the items themselves, use the key specified in the item.

## 8.13 Result event list fragment

Several calls query a set of event objects (e.g. EPG information). These calls always return part of a result, too. Therefore, a fragment object is used, like with the item list, e.g:

```
<ltv:ResultEventFragment sequenceNumber="9869569843634-cghdfgh-27346"
    totalResults="14025" returnedResults="3" startIndex="9450">
  <ltv:ResultEventReference
    uuid="lkjdfksdfgsdfg"
    itemUuid="chl0://lo-00-8d-1c-4a-35-69-0146-8976246863667"/>
  <ltv:ResultEventReference
    uuid="lk56dfksdfgsdfg"
    itemUuid="chl0://lo-00-8d-1c-4a-35-69-0146-8976246863667"/>
  <ltv:ResultEventReference
    uuid="lk56dfk345gsdfg"
    itemUuid="chl0://lo-00-8d-1c-3a-25-23-1245-1564790863345"/>
</ltv:ResultEventFragment>
```

Extra call are required to receive the actual information inside.

# 9 Open API methods

## 9.1 Connecting

Technically, the remote device is not connected to the TV. However, the device will authenticate itself to the TV in a simple challenge/response call. Depending on the security level requested, different challenge/response algorithms are be used.

### 9.1.1 Pairing

This public call introduces the client to the TV. The TV may or may not ask the end user to accept the device. The TV may grant the mobile device access to it.

Because of the possible length of a user interaction, this command is meant to have a finite life-time, the TV is allowed not to answer at all (preventing DoS), or to declare the query as "Accepted", "Denied", "Pending", "Full".

However, for the public data API, no security handshaking is required. The client simple provides a short device description (max. 40 characters) and a fcid. It receives in turn a client id valid for some time. At any time, the TV can cancel the "logical connection" associated with the client id (because of the lack of resources). In that case, the client has to re-request the access (the TV) might have cached the user's response.

```
<ltv:RequestAccess>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>?</ltv:ClientId>
  <ltv:DeviceType>Apple iPad</ltv:DeviceType>
  <ltv:DeviceName>Olli's iPad</ltv:DeviceName>
  <ltv:DeviceUUID>28:6A:BA:29:F7:06</ltv:DeviceUUID>
  <ltv:RequesterName>Assist Media App</ltv:RequesterName>
</ltv:RequestAccess>
```

might yield the following response:

```
<ltv:RequestAccessResponse>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:AccessStatus>Pending</ltv:AccessStatus>
</ltv:RequestAccessResponse>
```

The fields are as follows (for examples see above):

- *DeviceType* – should be the type of the device requesting access
- *DeviceName* – the name of the specific device
- *DeviceUUID* – unique Id of that device. Can be any string but it is strongly recommended to use the MAC-Address in standard notation as above
- *RequesterName* – name of the (part of the) application requesting access. This is meant to be used to differentiate between multiple applications on the same device requesting access of even different users of the same app. This could i.e. look like “Assist Media App:0” for the first user or “Assist Media App:1” for the second and so on.

The next request might look like, to reidentify:

```
<ltv:RequestAccess>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:DeviceType>Apple iPad</ltv:DeviceType>
  <ltv:DeviceName>Olli's iPad</ltv:DeviceName>
  <ltv:DeviceUUID>28:6A:BA:29:F7:06</ltv:DeviceUUID>
  <ltv:RequesterName>Assist Media App</ltv:RequesterName>
</ltv:RequestAccess>
```

And might yield the following response:

```
<ltv:RequestAccessResponse>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:AccessStatus>Denied</ltv:AccessStatus>
</ltv:RequestAccessResponse>
```

Or, maybe:

```
<ltv:RequestAccessResponse>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:AccessStatus>Accepted</ltv:AccessStatus>
</ltv:RequestAccessResponse>
```

Please do no spam the TV with requests, one request is by definition answered within two seconds.

Most requests will work only if a client id is provided. Depending on the target's complexity, it also might answer requests without client id. Is is not the purpose of the handshaking to replace secure transport media like ssl.

## 9.2 Retrieving Device Information

This call allows the client to retrieve additional information from the TV regarding its configuration.

GetDeviceData can be used for discovery. The value of element ClientId can be left empty, there is no need to call RequestAccess first.

```
<ltv:GetDeviceData>
  <ltv:fcid>8138942</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
</ltv:GetDeviceData>
```

might yield the following response:

```
<ltv:GetDeviceDataResponse>
  <ltv:fcid>8138942</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Chassis>SL150</ltv:Chassis>
  <ltv:SW-Version>PV7.2.1</ltv:SW-Version>
  <ltv:MAC-Address>00:0a:0b:0c:0d:0e</ltv:MAC-Address>
  <ltv:MAC-Address-LAN>34:f6:2d:ff:ff:ff</ltv:MAC-Address-LAN>
  <ltv:MAC-Address-WLAN/>
  <ltv:Location>Germany</ltv:Location>
  <ltv:NetworkHostName>hl1</ltv:NetworkHostName>
  <ltv:StreamingServerName>Remote TV</ltv:StreamingServerName>
  <ltv:OwnVolumeId>DLN1://2fd9d370-1dd2-11b2-822f-00098219b9bb</ltv:OwnVolumeId>
</ltv:GetDeviceDataResponse>
```

Some elements of the response might not be supported by older chassis.

For SL220, from PV1.9.1 on also the MAC address is returned as above to support i.e. wake on (w)lan functionality. In addition to the MAC-Address of the currently active network device, MAC-Address-LAN and MAC-Address-WLAN are returned. MAC-Address-WLAN may be empty, when there is no wireless hardware installed.

Location returns the location of the device, which is set at initial installation.

If there is some kind of error while collecting the device data, the response will contain empty strings for some or all values:

```
<ltv:GetDeviceDataResponse>
  <ltv:fcid>8138942</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Chassis/>
  <ltv:SW-Version/>
  <ltv:MAC-Address/>
  <ltv:MAC-Address-LAN/>
  <ltv:MAC-Address-WLAN/>
  <ltv:Location/>
  <ltv:NetworkHostName/>
  <ltv:StreamingServerName/>
  <ltv:OwnVolumeId/>
</ltv:GetDeviceDataResponse>
```

## 9.3 Subscribing

### 9.3.1 Subscribe

A client most will be interested in particular actions happening on the TV. Therefore the client can subscribe to receive notifications for particular changes happening.

```
<wse:Subscribe>
  <wse:Delivery>
    <wse:NotifyTo>
      <wsa:Address>
        soap.udp://192.168.0.15:21436/udp-sink/192.168.0.13
      </wsa:Address>
      <wsa:ReferenceProperties>
      </wsa:ReferenceProperties>
    </wse:NotifyTo>
  </wse:Delivery>
</wse:Subscribe>
```

This message, when sent to the appropriate event source, adds a new subscriber to this event source. The subscriber will be ready to receive events using the given (arbitrary) address.

Note: The current implementation will only support SOAP over UDP notifications.

@@@ insert event example here @@@

## 9.4 Injecting

Injecting commands certain devices always has undefined semantics: The TV may or may not be in the expected state, the command might interfer with some action started by the TV or another user. Also, by using the term injection, we refer to actions that naturally have no direct result or acknowledge.

So the TV might respond with an error, if the query is wrong by itself, but it cannot grant any certain behaviour or successful operation due to the requirement's nature.

### 9.4.1 Inject RC event

To emulate legacy behaviour, it is possible to inject infrared key codes. Please note, that IR key codes consist of PRESS, REPEAT, DELAYED and RELEASE codes.

You have to take care to properly send release codes! To optimize behaviour, several key codes can be combined in one SOAP message.

Note: Because of the design of current LOEWE software, key events cannot be acknowledged, and successful delivery cannot be granted.

```
<ltv:InjectRCKey>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:InputEventSequence>
    <ltv:RCKeyEvent alphabet="l2700" mode="press" value=”1”/>
    <ltv:RCKeyEvent alphabet="l2700" mode="release" value=”1”/>
  </ltv:InputEventSequence>
</ltv:InjectRCKey>
```

In order to properly simulate a long keypress you should generate (at least) one “repeat” event between press and release and one “delayed” event.

The method call may generate an error response. The alphabet selects the desired IR meaning. The value must be a numeric value valid for the chosen alphabet. Symbolic values today are not supported. The resulting actions on the TV from this query are not defined!

Note: The numeric alphabet values might get extended if they are found to be insufficient. Currently supported key code values can be found in the following subsections.

#### 9.4.1.1 Key codes for alphabet l2700

Alphabet “l2700” covers key codes for Loewe remote control units.

| Key | Value | Key | Value |
|---|---|---|---|
| 0 | 0 | ON_OFF | 12 |
| 1 | 1 | MUTE | 13 |
| 2 | 2 | EPG | 15 |
| 3 | 3 | RIGHT | 16 |
| 4 | 4 | LEFT | 17 |
| 5 | 5 | VOL_MI | 20 |
| 6 | 6 | VOL_PL | 21 |
| 7 | 7 | TV_ON | 22 |
| 8 | 8 | PROG_MI | 23 |
| 9 | 9 | PROG_PL | 24 |
| PIP | 10 | TV_OFF | 25 |
| MENU | 11 | GREEN | 26 |
| RED | 27 | ASPECT | 90 |

| Key | Value | Key | Value |
|---|---|---|---|
| UP | 32 | TIMER | 91 |
| DOWN | 33 | DR_ARCHIVE | 92 |
| PIC | 35 | AV1 | 114 |
| OK | 38 | AV2 | 115 |
| BLUE | 40 | AV3 | 116 |
| YELLOW | 43 | AVS | 117 |
| MEDIA | 49 | VGA | 118 |
| RADIO | 53 | HDMI1 | 119 |
| TTX | 60 | COMP | 120 |
| END | 63 | HDMI2 | 121 |
| SOUND | 64 | HDMI3 | 122 |
| BACK | 65 | HDMI4 | 123 |
| TV_MODE_ON | 72 | VIDEO | 124 |
| RADIO_MODE_ON | 73 | SPDIF_IN | 125 |
| EPG_STAR | 78 | | |
| INFO | 79 | | |
| PIP_HASH | 88 | | |

#### 9.4.1.2 Key codes for alphabet l2700-hdr

Alphabet “l2700-hdr” covers key codes for Loewe remote control units related to recording and player control.

| Key | Value | Key | Value |
|---|---|---|---|
| HDR_PAUSE | 41 | HDR_PLAY | 53 |
| HDR_REW | 50 | HDR_STOP | 54 |
| HDR_FF | 52 | HDR_REC | 55 |

#### 9.4.1.3 Key codes for alphabet sharp-le65

Alphabet “sharp-le65” includes key codes which are unique to the Sharp remote control units.

| Key | Value | Key | Value |
|---|---|---|---|
| 3D | 0 | FREEZE_HOLD_TXT | 5 |
| ATV | 1 | SAT | 6 |
| DTV | 2 | SUBTITLE | 7 |
| FLASHBACK | 3 | REVEAL_HIDDEN_TXT | 8 |
| ECO | 4 | SUBPAGE | 9 |

### 9.4.2 Inject Keyboard event

Apart from RC keys, it is also possible to inject keyboard keys, either by scancode or by unicode value.

Note: Because of the design of current LOEWE software, key events cannot be acknoledged, and successful delivery cannot be granted.

```
<ltv:InjectKeyboardKey>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:InputEventSequence>
    <ltv:KeyboardKeyEvent mode=”press” modifiers=”scancode” value=”18”/>
    <ltv:KeyboardKeyEvent mode=”release” modifiers=”scancode” value=”18”/>
    <ltv:KeyboardKeyEvent mode=”press” modifiers=”” value=”101”/>
    <ltv:KeyboardKeyEvent mode=”release” modifiers=”” value=”101”/>
  </ltv:InputEventSequence>
</ltv:InjectKeyboardKey>
```

The *mode* attribute is just the same as for RC keys, i.e. one of ( press, release, repeat ). The *modifiers* are a comma seperated list of ( alt, ctrl, shift, scancode ). These currently only make sense if used in conjuction with “scancode”.
Furthermode, scancodes always refer to a german keyboard layout at the moment and should only be used to transfer special keys, such as F1-F12 and so on. The *value* then contains either the scancode or unicode value of the key to inject, depending upon the presence of “*scancode*” in the modifiers. For all “normal” keys, passing unicode values is strongly encouraged.

### 9.4.3 Wake On LAN

It is also possible to wake a tv set from network standby using these commands.
In order to do this, send an RC key event for the “TV” key (L2700 keycode 22) as follows:

```
<ltv:InjectRCKey>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:InputEventSequence>
    <ltv:RCKeyEvent alphabet="l2700" mode="press" value=”22”/>
    <ltv:RCKeyEvent alphabet="l2700" mode="release" value=”22”/>
  </ltv:InputEventSequence>
</ltv:InjectRCKey>
```

### 9.4.4 Local control

@@@ refer to injection of input events. @@@

### 9.4.5 Volume control

Volume control is supported. This will ask the TV to modify its local output. What local output exactly refers to depends on the TVs configuration. In certain configurations, this may not change anything (external amplifiers).

```
<ltv:SetVolume>
    <fcid>18725463</fcid>
    <ClientId>LRemoteClient-0-1314017969</ClientId>
    <Value>500000</Value>
</ltv:SetVolume>
```

The *value* is in range 0 – 999999. It is also possible to ask for the current volume level:

```
<ltv:GetVolume>
    <fcid>18725463</fcid>
    <ClientId>LRemoteClient-0-1314017969</ClientId>
</ltv:GetVolume>
```

The requests are answered with a *SetVolumeResponse* or *GetVolumeResponse* respectively, both of which have the same fields as the *SetVolume* request.

### 9.4.6 Mute control

Analogous to volume control, it is also possible to get or set the mute status of the main speakers:

```
<ltv:SetMute>
    <fcid>18725463</fcid>
    <ClientId>LRemoteClient-0-1314017969</ClientId>
    <Value>1</Value>
</ltv:SetMute>
```

The *value* is in range 0 – 1 with 1 meaning muted and 0 meaning not muted. Also querying is supported, as before:

```
<ltv:GetMute>
    <fcid>18725463</fcid>
    <ClientId>LRemoteClient-0-1314017969</ClientId>
</ltv:GetMute>
```

The requests are answered with a *SetMuteResponse* or *GetMuteResponse* respectively, both of which have the same fields as the *SetMute* request.

## 9.5 About media items

### 9.5.1 Get Media Items

This public call retrieves a standard set of information about a single item. Technically spoken, it returns the media information that would be associated with the media item if you would have started playback at the time of the query. Note, that not all directory services are able to provide all of the available information without playing this media item.

```
<urn:GetMediaItem>
  <fcid>18274536</fcid>
  <ClientId>LRemoteClient-0-1406788706</ClientId>
  <MediaItemReference mediaItemUuid="chl0:00-00040001ffff044d6dca"/>
</urn:GetMediaItem>
```

might yield the following response:

```
<m:GetMediaItemResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>18274536</m:fcid>
  <m:ClientId>LRemoteClient-0-1406788706</m:ClientId>
    <m:ResultItem>
      <m:MediaItem itemInfoClass="object.item.videoItem">
        <m:uuid>chl0:00-00040001ffff044d6dca</m:uuid>
        <m:AncestorUuid>18446744073709551615</m:AncestorUuid>
        <m:Locator>channel://1:sl5a2eb1a7-6687-4a0e-8f32-af2c82f0b6eb</m:Locator>
        <m:Caption>1</m:Caption>
      </m:MediaItem>
      <m:MediaInformation>
        <m:ShortInfo>Das Erste</m:ShortInfo>
        <m:Attributes>131080</m:Attributes>
        <m:StreamingUrl>http://192.168.10.141:1543/lt0/0/$1$sl27cc5cda-ec69-4cde-abd9-
0a8bd0f9f8bf$fffba1671e70483aa92</m:StreamingUrl>
      </m:MediaInformation>
    </m:ResultItem>
</m:GetMediaItemResponse>
```

The field Attributes describes an Integer bitmask in decimal notation of attributes set for this media item:

| | |
|---|---|
| 1 | Encrypted station |
| 2 | CI+ station |
| 4 | HD station |
| 8 | EPG data acquisition for this station |
| 16 | Parental lock for this station |
| 32 | Gaming mode active for this station |
| 64 | HBBTV start behavior for this station |
| 128 | New found services after an automatic scan or DCM |
| 256 | Marked invalid, when not found after a channel search |
| 512 | Marked as not erased |
| 1024 | Locked (obsolete) |
| 2048 | Assigned CI slot (DVB services only) |
| 4096 | TTX preview page (analog services only) |

| | |
|---|---|
| 8192 | TTX subtitle page (analog services only) |
| 16384 | TTX character encryption (analog services only) |
| 32768 | Indication that a service has LCN conflict |
| 65536 | Indication that a new service (from last scan) has LCN conflict |
| 131072 | Streaming is possible (Only return value, not selectable) |
| 262144 | Protected use for hotel mode only |
| 524288 | Service temporarily not available (e.g. due to dual record. restriction) |

## 9.6 About media events

### 9.6.1 Get media events

This public call retrieves a standard set of information about a media event. The actual amount of information varies depending on the media directory involved.

```
<ltv:GetMediaEvent>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediaEventReference
    mediaEventUuid="chl://lo-00-8d-1c-4a-35-69-0146-8976246863667"/>
</ltv:GetMediaEvent>
```

might yield the following response:

```
<ltv:GetMediaEventResponse>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:MediaEvent>
    <ltv:uuid>chl://lo-00-8d-1c-4a-35-69-0146-8976246863667</ltv:uuid>
    <ltv:MediaItemReference mediaItemUuid="01-345-14-12-ab8"/>
    <ltv:MediaInformation>
      <ltv:ShortInfo>heute Journal</ltv:ShortInfo>
      <ltv:ExtendedInfo>Aktuelles vom Tage.</ltv:ExtendedInfo>
    </ltv:MediaInformation>
    <ltv:Availability startTime="16741641232" duration="1200"/>
  </ltv:MediaEvent>
</ltv:GetMediaEventResponse>
```

It is perfectly OK to put several key identifiers into the query. However, it is neither guaranteed for the answer to contain the items in the same order, nor to be contained in only one GetMediaEventResponse at all. The sl TV may distribute the information parts to any number of replies containing any number of items in any order. It is up to the client to identify the items.

## 9.7 Channel lists

### 9.7.1 Get channel list

This public get channel query allows a device to retrieve a part of the channel list. Find below an example query for the channel list.

```
<ltv:GetChannelList>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ChannelListView>default</ltv:ChannelListView>
  <ltv:QueryParameters>
    <ltv:Range startIndex="100" maxItems="100"/>
    <ltv:OrderField field="userChannelNumber" type="ascending"/>
  </ltv:QueryParameters>
</ltv:GetChannelList>
```

The query above may result in a response which might contain a result like below. Please note that the TV is free to deliver less entries. or entries at different positions. Also, the TV may split the answers into several different calls.

```
<ltv:GetChannelListResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ChannelListView>default</ltv:ChannelListView>
  <ltv:ChannelListName/>
  <ltv:ResultItemFragment sequenceNumber="9869569843634-cghdfgh-27346"
      totalResults="100" returnedResults="2" startIndex="100" hash="1163108926">
    <ltv:ResultItemReference
     mediaItemUuid="lo-00-8d-1c-4a-35-69-0146-8976246863667"/>
    <ltv:ResultItemReference
     mediaItemUuid="lo-00-8d-1c-3a-25-23-1245-1564790863345"/>
  </ltv:ResultItemFragment>
</ltv:GetChannelListResponse>
```

The actual items can be resolved using the GetMediaItem call.

Please note, that the GetChannelList call only is a special case of the SearchMedia call.

The ChannelListView parameter specifies which list to query. “default” means the global channel list. As of this writing, for SL1xx it is also possible to query the favourite lists by passing “favlist0” … “favlist5” as ChannelListView. For SL2xx it will remain similar, yet the number of available favourite lists might change.
In the answer the “ChannelListName” property is used to communicate the names of favourite lists. If the default view was queried, this will be an empty tag, like in the example above.

The view name for a list does not change while the TV is running.

### 9.7.2 Get list of channel lists

This public get channel query allows a client to retrieve a part of the list of channel lists, which can be either user-defined favourite lists or network based lists (e.g.  DVB-T).

The setting of the QueryParameters allows control of the length of the server response by the client. In the following example, the client requests the list from the very first entry (startIndex=”0”, best practice for first query) and limits the number of results to 5 (maxItems=”5”). If the response by the server indicates that more than 5 results are available, the client can increment the index and query the next block of list results. Parameter OrderField is not evaluated at the moment.

The AdditionalParameters allow a query just for the active list, when attribute flag isActiveList is set to value 1.

```
<ltv:GetListOfChannelLists>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:QueryParameters>
    <ltv:Range startIndex="0" maxItems="5"/>
    <ltv:OrderField field="userChannelNumber" type="ascending"/>
  </ltv:QueryParameters>
  <ltv:AdditionalParameters>
    <ltv:Properties isActiveList="0"/>
  </ltv:AdditionalParameters>
</ltv:GetListOfChannelLists>
```

The response to the query might look like the following.

```
<ltv:GetListOfChannelListsResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ResultItemChannelLists sequenceNumber="9076676" totalResults="4" returnedResults="4"
startIndex="0">
    <ltv:ResultItemChannelList>
      <ltv:View>avlistsl253922f2-c717-4f3c-bb2d-7d7b4fc8ace7</ltv:View>
      <ltv:Name>#3051</ltv:Name>
      <ltv:TotalResults>9</ltv:TotalResults>
      <ltv:Hash>3546968615</ltv:Hash>
    </ltv:ResultItemChannelList>
    <ltv:ResultItemChannelList>
      <ltv:View>fastscansl961096e5-f508-45a1-8dec-4865bfb9ef9b</ltv:View>
      <ltv:Name>ASTRA1 19,2°E</ltv:Name>
      <ltv:TotalResults>454</ltv:TotalResults>
      <ltv:Hash>754331397</ltv:Hash>
    </ltv:ResultItemChannelList>
    <ltv:ResultItemChannelList>
      <ltv:View>favlistsl507dcd40-0530-4ae7-bf2c-d898749d0440</ltv:View>
      <ltv:Name>Selected stations</ltv:Name>
      <ltv:TotalResults>15</ltv:TotalResults>
      <ltv:Hash>3365940522</ltv:Hash>
    </ltv:ResultItemChannelList>
    <ltv:ResultItemChannelList>
      <ltv:View>favlistsl3e5518e8-72f7-41dc-b279-99256f39c62d</ltv:View>
      <ltv:Name>Personal list 1</ltv:Name>
      <ltv:TotalResults>99</ltv:TotalResults>
      <ltv:Hash>521398698</ltv:Hash>
    </ltv:ResultItemChannelList>
  </ltv:ResultItemChannelLists>
  <ltv:Result>OK</ltv:Result>
</ltv:GetListOfChannelListsResponse>
```

Parameter sequenceNumber can be used to put partial server responses in the correct sequence. Parameter totalResults denotes the number of all lists at the time of the query. Parameter returnedResults denotes the number of list results in the current server response, which can be less than the queried number of lists defined by query parameter maxItems, if less are available.

Each result item ResultItemChannelList combines the internal view name (prefix fastscan for network lists, favlist for favourite lists, avlist for AV lists and the UUID of the list) with the actual channel name, the total number of channel items and a hash value. This information can be used to query a number of channel items of a list by method GetChannelList.

The hash value changes when a list receives an update. These updates can have several reasons, e.g. a service has been added to a list or a personal list has been added. A client should use this as an indication to update lists with changes via method GetChannelList.

The view name for a list does not change while the TV is running.

The Result parameter can have values "OK", "SYNC" or "NONE", if no channel lists have been found. "SYNC" does signal an ongoing, internal list synchronization, so that a client can try to request list data again at a later point.

A list of virtual lists can be obtained by separate call GetListOfVirtualLists.

### 9.7.3 Add personal channel list

Add an empty personal channel list. Parameters are the name of the list and type, which is described by an attribute value. Supported attribute values are “4” for a TV list and “8” for radio list.

Adding a personal channel list invalidates the view name mapping. You have to call function GetListOfChannelLists again to get a valid view name mapping.

```
<ltv:AddFavoriteList>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Name>MyNewList</ltv:Name>
  <ltv:AdditionalAttributes>4</ltv:AdditionalAttributes>
</ltv:AddFavoriteList>
```

The response to the query might look like the following. Result can be either “OK” or “KO”.

```
<ltv:AddFavoriteListResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:AddFavoriteListResponse>
```

### 9.7.4 Remove personal channel list

Remove an existing personal channel list. Parameter is the view name of the list.

Removing a personal channel list invalidates the view name mapping. You have to call function GetListOfChannelLists again to get a valid view name mapping.

```
<ltv:RemoveFavoriteList>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ViewName>favlist0</ltv:ViewName>
</ltv:RemoveFavoriteList>
```

The response to the query might look like the following. Result can be either “OK” or “KO”.

```
<ltv:RemoveFavoriteListResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:RemoveFavoriteListResponse>
```

### 9.7.5 Rename personal channel list

Rename an existing personal channel list. Parameters are the view name of the list and its new name.

```
<ltv:RenameFavoriteList>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ListViewName>favlist0</ltv:ListViewName>
  <ltv:NewName>MyOtherNewList</ltv:NewName>
</ltv:RenameFavoriteList>
```

The response to the query might look like the following. Result can be either “OK” or “KO”.

```
<ltv:RenameFavoriteListResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:RenameFavoriteListResponse>
```

### 9.7.6 Add services to a personal channel list

Append services to the end of a personal channel list. The services are identified by its Locator, which can be retrieved by function GetMediaItem. The order of the new services in the list will be the same as the Locators in this call.

Parameters are the view name of the list and a LocatorSequence of Locators.

Adding services to a personal channel list invalidates the number of result items returned by GetChannelList and GetListOfChannelLists. You have to call these functions again to get the correct number of services in a list.

```
<ltv:AddServicesFavoriteList>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ViewName>favlist0</ltv:ViewName>
  <ltv:LocatorSequence>
    <ltv:Locator Locator="channel://1:fff7a1671e77e03cebd"/>
    <ltv:Locator Locator="channel://13:fffba1671e7000376ca"/>
  </ltv:LocatorSequence>
</ltv:AddServicesFavoriteList>
```

The response to the query might look like the following. Result can be either “OK” or “KO”.

```
<ltv:AddServicesFavoriteListResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:AddServicesFavoriteListResponse>
```

### 9.7.7 Remove services from a personal channel list

Remove a block of services from a personal channel list. A block can be only one service or a number of consecutive services. The block is identified by the channel number of the first and last service (both inclusive) in the personal channel list.

Parameters are the view name of the list and the channel numbers for the first and last service of the block.

It is recommended that the channel number of the last service of a block has to be greater than or equal to the channel number of the first block. Removed services are marked as deleted and will remain present in other channel lists.

Removing services from a personal channel list invalidates the number of result items returned by GetChannelList and GetListOfChannelLists. You have to call these functions again to get the correct number of services in a list.

```
<ltv:RemoveServicesFavoriteList>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ViewName>favlist0</ltv:ViewName>
  <ltv:NumBegin>1</ltv:NumBegin>
  <ltv:NumEnd>2</ltv:NumEnd>
</ltv:RemoveServicesFavoriteList>
```

The response to the query might look like the following. Result can be either “OK” or “KO”.

```
<ltv:RemoveServicesFavoriteListResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:RemoveServicesFavoriteListResponse>
```

### 9.7.8 Move services in a personal channel list

Move a block of services in an existing personal channel list. A block can be only one service or a number of consecutive services. The block is identified by the channel number of the first and last service (both inclusive) in the personal channel list.

Parameters are the view name of the list, the channel numbers for the first and last service of the block and the channel number, after which this block shall be moved.

It is recommended that the channel number of the last service of a block has to be greater than or equal to the channel number of the first block.

Moving services in a personal channel list invalidates the order of result items returned by GetChannelList. You have to call this function again to get the correct order of services in a list.

```
<ltv:MoveServicesFavoriteList>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ViewName>favlist0</ltv:ViewName>
  <ltv:NumBegin>1</ltv:NumBegin>
  <ltv:NumEnd>1</ltv:NumEnd>
  <ltv:NumNewPosition>4</ltv:NumNewPosition>
</ltv:MoveServicesFavoriteList>
```

The response to the query might look like the following. Result can be either “OK” or “KO”.

```
<ltv:MoveServicesFavoriteListResponse>
  <ltv:fcid>4138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:MoveServicesFavoriteListResponse>
```

### 9.7.9 Import service lists

Import XML data, to clone all services and service lists from another TV. In addition DVB network settings are cloned too, if available.

The XML data is embedded by element Data. This data can become quite huge, in the following example query some additional elements of the same type have been removed, see comments.

```
<ltv:ImportServiceLists>
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1426235174</ltv:ClientId>
  <ltv:Data>
        <?xml-stylesheet type="text/xsl" href="servicelist.xsl"?>
        <servicelist version="1.9.9" db="sqlite3" scheme="43" release="7">
          <tuners>
            <tuner Frontend="8" NITId="-1" NITVersion="-1" Oid="1" Satellite="4" Tid="1019" TunerId="1"
TunerUuid="fff7a1671e77e0"/>
            <!--tuner elements removed-->
            <dvbs-tuner Band="2" CodeRate="7" Frequency="12187218" Modulation="2" Polarization="2"
SatelliteId="4" SpectrumInversion="1" SymbolRate="27500" TunerId="3"/>
            <!--dvbs-tuner elements removed-->
            <dvbs2-tuner Band="1" FrameLength="0" Frequency="11494075" ModulationCoding="13" Pilots="1"
Polarization="2" SatelliteId="4" SpectrumInversion="1" SymbolRate="21999" TunerId="1"/>
            <!--dvbs2-tuner elements removed-->
          </tuners>
          <services>
            <service CreationDate="0" Encrypted="0" EngineAutoStart="1" EpgService="1" EventInfo="0"
GamingMode="0" Hdtv="1" IsReceivable="1" MediaType="1" ModificationDate="0" Name="Das Erste HD"
ParentalLock="0" Pid="9" Selectable="1" ShortName="" SoundSelection="0" ThumbUrl="" Type="8" Uri=""
Uuid="fff7a1671e77e03cebd" Visible="1" VolumeCorrection="0"/>
            <!--service elements removed-->
            <dvb-service CiPlus="0" CiPlusBrandIds="" CiPlusFreeCiMode="0" CiPlusTimestamp="1426240875"
CiSlotConfig="0" DefaultAuth="" DefaultAuthPrio="0" ExternalData="0" ServiceId="9" Sid="10301"
TtxCharEnc="0" TtxPreviewPage="301" TtxSubtitlePage="150" TunerId="1" VirtualSid="-1"/>
            <!--dvb-service elements removed-->
          </services>
          <favorites>
            <favorite-list Attributes="0" BATId="-1" BATVersion="-1" CreationDate="0"
Creator="servicelist://loewe.ASTRA1 19,2°E" MaxChannelNumber="132" ModificationDate="0" NITId="-1"
NITVersion="-1" Name="ASTRA1 19,2°E" NetworkId="-1" Pid="3" SDTVersion="-1" Uuid="sl1ac32476-aa9c-
442e-b278-8ee06bde89fd"/>
            <!--favorite-list elements removed-->
            <favorite-item Active="0" Attribute="0" ChannelNum="1" FavoriteId="2" Id="1" OriginalFavoriteId="2"
Selectable="-1" ServiceId="1" ServiceName="" Visible="-1"/>
            <!--favorite-item elements removed-->
            <lcn FavoriteId="2" Id="1" Lcn="1" ServiceId="1"/>
            <!--lcn elements removed-->
          </favorites>
        </servicelist>
        <OpSettings>
          <OperatroProfile>
            <OpId>00000000.0085</OpId>
            <BarkerChannel>
              <SID>3</SID>
              <TSID>2</TSID>
              <ONID>134</ONID>
            </BarkerChannel>
          </OperatroProfile>
        </OpSettings>
  </ltv:Data>
</ltv:ImportServiceLists>
```

```
<m:ImportServiceListsResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1426235174</m:ClientId>
  <m:Result>OK</m:Result>
</m:ImportServiceListsResponse>
```

### 9.7.10 Export service lists

Export XML data, which can be used to clone all services and service lists to another TV. In addition DVB network settings are cloned too, if available.

```
<ltv:ExportServiceLists>
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1426235174</ltv:ClientId>
</ltv:ExportServiceLists>
```

The XML data is embedded in the response by element Data. This data can become quite huge, in the following example response some additional elements of the same type have been removed, see comments.

```
<ltv:ExportServiceListsResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <ltv:fcid>?</ltv:fcid>
    <ltv:ClientId>LRemoteClient-0-1426235174</ltv:ClientId>
      <ltv:Data>
        <?xml-stylesheet type="text/xsl" href="servicelist.xsl"?>
        <servicelist version="1.9.9" db="sqlite3" scheme="43" release="7">
          <tuners>
            <tuner Frontend="8" NITId="-1" NITVersion="-1" Oid="1" Satellite="4" Tid="1019" TunerId="1"
TunerUuid="fff7a1671e77e0"/>
            <!--tuner elements removed-->
            <dvbs-tuner Band="2" CodeRate="7" Frequency="12187218" Modulation="2" Polarization="2"
SatelliteId="4" SpectrumInversion="1" SymbolRate="27500" TunerId="3"/>
            <!--dvbs-tuner elements removed-->
            <dvbs2-tuner Band="1" FrameLength="0" Frequency="11494075" ModulationCoding="13" Pilots="1"
Polarization="2" SatelliteId="4" SpectrumInversion="1" SymbolRate="21999" TunerId="1"/>
            <!--dvbs2-tuner elements removed-->
          </tuners>
          <services>
            <service CreationDate="0" Encrypted="0" EngineAutoStart="1" EpgService="1" EventInfo="0"
GamingMode="0" Hdtv="1" IsReceivable="1" MediaType="1" ModificationDate="0" Name="Das Erste HD"
ParentalLock="0" Pid="9" Selectable="1" ShortName="" SoundSelection="0" ThumbUrl="" Type="8" Uri=""
Uuid="fff7a1671e77e03cebd" Visible="1" VolumeCorrection="0"/>
            <!--service elements removed-->
            <dvb-service CiPlus="0" CiPlusBrandIds="" CiPlusFreeCiMode="0" CiPlusTimestamp="1426240875"
CiSlotConfig="0" DefaultAuth="" DefaultAuthPrio="0" ExternalData="0" ServiceId="9" Sid="10301"
TtxCharEnc="0" TtxPreviewPage="301" TtxSubtitlePage="150" TunerId="1" VirtualSid="-1"/>
            <!--dvb-service elements removed-->
          </services>
          <favorites>
            <favorite-list Attributes="0" BATId="-1" BATVersion="-1" CreationDate="0"
Creator="servicelist://loewe.ASTRA1 19,2°E" MaxChannelNumber="132" ModificationDate="0" NITId="-1"
NITVersion="-1" Name="ASTRA1 19,2°E" NetworkId="-1" Pid="3" SDTVersion="-1" Uuid="sl1ac32476-aa9c-
442e-b278-8ee06bde89fd"/>
            <!--favorite-list elements removed-->
            <favorite-item Active="0" Attribute="0" ChannelNum="1" FavoriteId="2" Id="1" OriginalFavoriteId="2"
Selectable="-1" ServiceId="1" ServiceName="" Visible="-1"/>
            <!--favorite-item elements removed-->
            <lcn FavoriteId="2" Id="1" Lcn="1" ServiceId="1"/>
            <!--lcn elements removed-->
          </favorites>
        </servicelist>
        <OpSettings>
          <OperatroProfile>
            <OpId>00000000.0085</OpId>
            <BarkerChannel>
              <SID>3</SID>
              <TSID>2</TSID>
              <ONID>134</ONID>
            </BarkerChannel>
          </OperatroProfile>
        </OpSettings>
    </ltv:Data>
  <ltv:Result>OK</ltv:Result>
</ltv:ExportServiceListsResponse>
```

### 9.7.11 Get list of virtual lists

Get the list of available virtual lists. A virtual list is a predefined list of services in a certain sort order, e.g. the sort order from a TV guide. The only information available for a virtual list are its view name and list name. To create an actual personal channel list from a virtual list, function AddFavoriteListFromVirtual can be called.

```
<ltv:GetListOfVirtualLists>
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-1-1429875201</ltv:ClientId>
  <ltv:QueryParameters>
    <ltv:Range startIndex="0" maxItems="50"/>
    <ltv:OrderField field="userChannelNumber" type="ascending"/>
  </ltv:QueryParameters>
</ltv:GetListOfVirtualLists>
```

The response to the query might look like the following.

The Result parameter can have values “OK”, “SYNC” or “NONE”, if no channel lists have been found. “SYNC” does signal an ongoing, internal list synchronization, so that a client can try to request list data again at a later point.

```
<m:GetListOfVirtualListsResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-1-1429875201</m:ClientId>
  <m:ResultItemChannelLists sequenceNumber="9076676" totalResults="5" returnedResults="5"
startIndex="0">
  <m:ResultItemChannelList>
    <m:View>virtuallistfile=tv_digital.xml&amp;creator=
    tv.magazine$tvdigital</m:View>
    <m:Name>TV Digital</m:Name>
    <m:TotalResults>0</m:TotalResults>
    <m:Hash>2356102164</m:Hash>
  </m:ResultItemChannelList>
  <m:ResultItemChannelList>
    <m:View>virtuallistfile=tv_direkt.xml&amp;creator=
    tv.magazine$tvdirekt</m:View>
    <m:Name>TV Direkt</m:Name>
    <m:TotalResults>0</m:TotalResults>
    <m:Hash>2969503783</m:Hash>
  </m:ResultItemChannelList>
  <m:ResultItemChannelList>
    <m:View>virtuallistfile=tv_hoerzu.xml&amp;creator=
    tv.magazine$tvhoerzu</m:View>
    <m:Name>Hörzu</m:Name>
    <m:TotalResults>0</m:TotalResults>
    <m:Hash>79605734</m:Hash>
  </m:ResultItemChannelList>
  <m:ResultItemChannelList>
    <m:View>virtuallistfile=tv_movie.xml&amp;creator=
    tv.magazine$tvmovie</m:View>
    <m:Name>TV Movie</m:Name>
    <m:TotalResults>0</m:TotalResults>
    <m:Hash>3393267761</m:Hash>
  </m:ResultItemChannelList>
  <m:ResultItemChannelList>
    <m:View>virtuallistfile=tv_spielfilm.xml&amp;creator=
    tv.magazine$tvspielfilm</m:View>
    <m:Name>TV Spielfilm</m:Name>
    <m:TotalResults>0</m:TotalResults>
    <m:Hash>2763753242</m:Hash>
  </m:ResultItemChannelList>
  </m:ResultItemChannelLists>
  <m:Result>OK</m:Result>
</m:GetListOfVirtualListsResponse>
```

### 9.7.12 Add personal channel lists from virtual

Add personal channel lists from virtual lists. Virtual lists can retrieved by GetListOfVirtualLists. These lists are identified by their view names “virtuallist”.

```
<urn:AddFavoriteListFromVirtual>
  <fcid>?</fcid>
  <ClientId>LRemoteClient-0-1430138065</ClientId>
  <ViewNameSequence>
    <ViewName ViewName="virtuallistfile=tv_digital.xml&amp;creator=
    tv.magazine$tvdigital"/>
    <ViewName ViewName="virtuallistfile=tv_direkt.xml&amp;creator=
    tv.magazine$tvdirekt"/>
    <ViewName ViewName="virtuallistfile=tv_hoerzu.xml&amp;creator=
    tv.magazine$tvhoerzu"/>
  </ViewNameSequence>
</urn:AddFavoriteListFromVirtual>
```

The response to the query might look like the following. Result can be either "OK" or "KO".

```
<ltv:AddFavoriteListFromVirtualResponse>
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1430138065</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:AddFavoriteListFromVirtualResponse>
```

## 9.8 Channel information

### 9.8.1 Get channel info

(deprecated, please use GetMediaItem)

### 9.9 Player control

In the TV device, various players are supported, like a main picture, a small picture etc. .

Every player can be associated with a medium. These are the requests to control the player. Please note that depending on medium type, player type, legal constraints associated with medium source, medium type, medium, intermediate ways of transport some playback controls may be disabled or pointless.

Before playback, a player must be associated with the medium. The player in turn might decide to pre-load parts of the medium, to initialize codecs or to ignore the request. Preparing a medium also may fail, e.g. if the medium type is not supported, or the medium may not be played back.

Once the medium is prepared, playback may start, standard playback controls may apply. Media also can be prepared in an “autostart” mode (if supported by the player), to enable gapless playback situations (usually desired for audio-only media). If autostart mode is enabled, the player automatically starts a medium that was prepared in the background, if the currently played medium terminates.

Playback may terminate automatically both regular or by an error, or by system control.

#### 9.9.1 Play Multiroom

Start synchronized playback of a live TV channel or a DR+ recording on up to three HL1 TVs. Target device for this query is the playback server, which initiates synchronized playback with up to two client devices.

Locator URL can either be a DR+ recording (drplus://) or a live TV channel (channel://). The Locator can be obtained from responses of GetDRPlusArchive for a DR+ recording and GetMediaItem for a live TV channel.

AncestorUuid has to be provided only for live TV channel playback and can be obtained from GetMediaItem.

MultiroomClientId attribute id can be obtained from GetDeviceData.

```
<ltv:PlayMultiroom>
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1425895334</ltv:ClientId>
  <ltv:Locator>drplus://UUID_HDR_0f60601b-dc63-482a-acdd-521526134cdf_00000014</ltv:Locator>
  <ltv:AncestorUuid>sl27cc5cda-ec69-4cde-abd9-0a8bd0f9f8bf</ltv:AncestorUuid>
  <ltv:MultiroomClientIds>
    <!--0 to 2 repetitions:-->
    <ltv:MultiroomClientId id="DLN1://2fd9d370-1dd2-11b2-822f-00098219b9bb"/>
  </ltv:MultiroomClientIds>
</ltv:PlayMultiroom>
```

```
<ltv:PlayMultiroomResponse>
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1425895334</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:PlayMultiroomResponse>
```

#### 9.9.2 Player status

Not implemented yet.

This is a common data described before other types at this point.

```
<ltv:PlayerStatus>
  <ltv:State>Playing</ltv:State>
  <ltv:Position>142.45</ltv:Position>
  <ltv:Speed>100</ltv:Speed>
</ltv:PlayerStatus>
```

State: Currently, one of

- Prepared
- Playing
- Stalled (buffering, pre-loading, system load)
- Pausing
- [Stopped still is reserved]

Position:

- If applicable, the position in seconds in the medium.

Speed:

- The current playback speed. Can also be negative.

### 9.9.3 Player Prepare

Not implemented yet.

```
<ltv:PlayerPrepare>
  <ltv:fcid>8138946</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Player>0</ltv:Player>
  <ltv:Locator>fsl2://ABC4-6292/Music/mysong.mp3</ltv:Locator>
</ltv:PlayerPrepare>
```

might yield the following response, including a medium id used in further communication:

```
<ltv:PlayerPrepareResult>
  <ltv:fcid>8138946</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerPrepareResult>
```

The subscription URL denotes the URL the user now implicitely is subscribed to. This URL can be used to explicitely unsubscribe the client.

### 9.9.4 Release Prepared

Not implemented yet.

To release a medium that current is prepared, use this command. If you don't release a medium it will consume superfluous resources, or it might playback accidentally.

```
<ltv:PlayerReleasePrepared>
  <ltv:fcid>8138943</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerReleasePrepared>
```

might yield the following response:

```
<ltv:PlayerReleasePreparedResult>
  <ltv:fcid>8138943</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
</ltv:PlayerReleasePreparedResult>
```

Note: It might happen that the medium cannot be released because it is no background medium any more. In that case, this call will fail, and and a regular medium remove call must be issued.

Note: After the medium has been released, the medium id is not valid any more.

Note: From a rights management perspective, the medium is in use already when prepared.

### 9.9.5 Play

Not implemented yet.

If a medium is not prepared in autostart mode, or if autostart cannot apply, a play request must be issued.

```
<ltv:PlayerPlay>
  <ltv:fcid>8138943</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerPlay>
```

Note: Reply to play is basically the same as for all other playback controls, it holds the current player status.

```
<ltv:PlayerPlayResult>
  <ltv:fcid>8138943</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
  <ltv:PlayerStatus>
    <ltv:State>Playing</ltv:State>
    <ltv:Position>142.45</ltv:Position>
    <ltv:Speed>100</ltv:Speed>
  </ltv:PlayerStatus>
</ltv:PlayerPlayResult>
```

### 9.9.6 Stop

Not implemented yet.

Stop the medium that currently is playing. If another medium is prepared in autostart mode, begin playback of the new medium.

```
<ltv:PlayerStop>
  <ltv:fcid>8138943</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerStop>
```

Note: Reply to stop is basically the same as for all other playback controls, it holds the current player status.

```
<ltv:PlayerStopResult>
  <ltv:fcid>8138943</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>nil</ltv:MediumId>
  <ltv:PlayerStatus>
    <ltv:State>Stopped</ltv:State>
    <ltv:Position>142.45</ltv:Position>
    <ltv:Speed>100</ltv:Speed>
  </ltv:PlayerStatus>
</ltv:PlayerStopResult>
```

### 9.9.7 Pause

Not implemented yet.

Suspend playback of the current medium.

```
<ltv:PlayerPause>
  <ltv:fcid>8138942</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerPause>
```

Result contains player status.

```
<ltv:PlayerPauseResult>
  <ltv:fcid>8138942</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
  <ltv:PlayerStatus>
    <ltv:State>Paused</ltv:State>
    <ltv:Position>142.45</ltv:Position>
    <ltv:Speed>100</ltv:Speed>
  </ltv:PlayerStatus>
</ltv:PlayerPauseResult>
```

### 9.9.8 Resume

Not implemented yet.

Resume playback of a formerly paused medium. Pause/resume does not change the playback speed.

```
<ltv:PlayerResume>
  <ltv:fcid>8138942</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerResume>
```

Result as always is the status.

```
<ltv:PlayerResumeResult>
  <ltv:fcid>8138942</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
  <ltv:PlayerStatus>
    <ltv:State>Playing</ltv:State>
    <ltv:Speed>100</ltv:Speed>
    <ltv:Position>142.45</ltv:Position>
  </ltv:PlayerStatus>
</ltv:PlayerResumeResult>
```

**Note**: that play is different from resume: Play starts the playback in the beginning, pause/resume are used later to suspend playback.

**Note**: For streams like live TV, internet radio etc., Resume may effectively result in skipping content.

### 9.9.9 Set Speed

Not implemented yet.

Define a playback speed for the medium. This is used to trigger special effects like slow motion, reverse playback.

Not all player/codec combinations are capable of rendering the medium at non-standard speeds. To implement winding use cases to the end used, you may have to fall back on a combination of seek to and set speed. Usually, only a small set of fixed playback speeds is available.

Setting a playback speed does not resume a paused medium.

```
<ltv:PlayerSetSpeed>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <lfv:Speed>400</ltv:Speed>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerSetSpeed>
```

Result as always is the status.

```
<ltv:PlayerSetSpeedResult>
  <ltv:fcid>8138941</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
  <ltv:PlayerStatus>
    <ltv:State>Playing</ltv:State>
    <ltv:Speed>500</ltv:Speed>
    <ltv:Position>172.45</ltv:Position>
  </ltv:PlayerStatus>
</ltv:PlayerSetSpeedResult>
```

**Note**: The requested speed can differ from the result, as not all players support all kind of speeds.

**Note**: The playback position cannot be computed from the starting playback position and the requested/resulting speed.

**Note**: The visual or sonic rendering from non-standard playback speeds is not defined. Please refer to output hints for further discussions.

### 9.9.10 Seek to

Not implemented yet.

Seek to a certain position in the medium. The position can be specified as an absolute (jump) target, or as a relative value.

```
<ltv:PlayerSeekTo>
  <ltv:fcid>8138940</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <lfv:AbsolutePosition>120</ltv:AbsolutePosition>
  <ltv:MediumId>7987987</ltv:MediumId>
</ltv:PlayerSeekTo>
```

Result as always is the status.

```
<ltv:PlayerSeekToResult>
  <ltv:fcid>8138940</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MediumId>7987987</ltv:MediumId>
  <ltv:PlayerStatus>
    <ltv:State>Playing</ltv:State>
    <ltv:Speed>100</ltv:Speed>
    <ltv:Position>121.451</ltv:Position>
  </ltv:PlayerStatus>
</ltv:PlayerSeekToResult>
```

The current player position can differ from the requested one depending on the player.

## 9.10 Changing channels

### 9.10.1 RC like zapping

The public API offers a method of channel switching equivalent to a classic remote control. The result of this zapping method is undefined because of the lack of any negotiation phase.

```
<ltv:ZapToMedia>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Player>0</ltv:Player>
  <ltv:Locator>chl://localhost/live/8976246863667</ltv:Locator>
</ltv:ZapToMedia>
```

will most likely generate a response like:

```
<ltv:ZapToMediaResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>0</ltv:Result>
</ltv:ZapToMediaResponse>
```

The “Result” field is an indicator whether the switching was successful. 0 denotes success. Keep in mind that this is an indicator only, and not fully reliable. For example you might receive a different other than 0 if the switching just takes too long, even though it is eventually successful.

The response might contain additional information.

### 9.10.2 Switching to another application

This API enables to switch to another application, i.e. the built-in web browser, opening it with a specific URL. Currently only the browser is supported for this message.

```
<ltv:ZapToApplication>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Application>browser</ltv:Application>
  <ltv:ContentURI>http://www.spiegel.de</ltv:ContentURI>
</ltv:ZapToApplication>
```

will most likely generate a response like:

```
<ltv:ZapToApplication>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>0</ltv:Result>
</ltv:ZapToApplication>
```

## 9.11 Current playback information

A client may wish to find out, what the tv currently displays on its screen. The GetCurrentPlayback method allows just that, within restrictions. Note that this functionality is preliminary and may be removed as soon as a proper eventing logic is in place.

```
<ltv:GetCurrentPlayback>
  <ltv:fcid>8138947</ltv.fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
</ltv:GetCurrentPlayback>
```

would yield the following response

```
<ltv:GetCurrentPlaybackResponse>
  <ltv:fcid>8138947</ltv.fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Mode>drplus</ltv:Mode>
  <ltv:Locator>drplus://10.10</ltv:Locator>
</ltv:GetCurrentPlaybackResponse>
```

*Mode* is currently one of tv, radio, drplus, browser, UNKNOWN. The *Locator* depends upon the *Mode*. For tv, radio and drplus you receive a *Locator* just like the ones you would use to “ZapToMedia”. For browser, the *Locator* is always empty, that is, not included. The same goes for internet radio: This will be communicated by the combination of *Mode*=radio and no *Locator*.

## 9.12 Event information

### 9.12.1 Get current event

The following query

```
<ltv:GetCurrentEvent>
  <ltv:fcid>8138947</ltv.fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Player>0</ltv:Player>
</ltv:GetCurrentEvent>
```

would yield the following response

```
<ltv:GetCurrentEventResponse>
  <ltv:fcid>8138947</ltv.fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ItemLocator>chl://localhost/live/1509764524575</ltv:ItemLocator>
  <ltv:MediaEvent>
    <ltv:MediaInformation>
      <ltv:ShortInfo>The River</ltv:ShortInfo>
      <ltv:ExtendedInfo>
        A journey into the depth of the Amazonas, exploring …
      </ltv:ExtendedInfo>
    </ltv:MediaInformation>
    <ltv:Availability>
      <ltv:ScheduledTime ltv:startTime="1235142089" duration="3600"/>
    </ltv:Availability>
  </ltv:MediaEvent>
</ltv:GetCurrentEventResponse>
```

This asks for the current event on the default main player (main screen), represented by the id zero. The query is associated with the (arbitrary) function call id 8138947.

The TV returns the event info of the current event, adding the function call id provided by the request, and giving the information that the media item carrying the event is distributed on has the item url "chl://localhost/live/1509764524575" in the running system.

### 9.12.2 Get next event

Basically the same as current event.

The following query

@@@ copy me @@@

would yield the following response

@@@ copy me @@@

This asks for the current event on the default main player (main screen), represented by the id zero. The query is associated with the (arbitrary) function call id 8138948.

The TV returns the event info of the current event, adding the function call id provided by the request.

## 9.13 Programming Timers

External applications may program record timers on the tv set:

```
<ltv:ProgramTimer>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Device>PVR</ltv:Device>
  <ltv:RecordingType>default</ltv:RecordingType>
  <ltv:Locator>
dvb://localhost/#?chlview=default&amp;progNum=2&amp;gcn=7&amp;onid=1&amp;tsid=1011&amp;sid=11110
  </ltv:Locator>
  <ltv:Timer starttime="1315494000" duration="3600" repeatDays="0"/>
</ltv:ProgramTimer>
```

“Device” specifies the device to record on. Currently only “PVR”, referring to the local DR+ recorder, is supported.

“RecordingType” will allow specifying different recording type i.e. for series recording etc. Currently only “default” is supported.

“Locator” is the usual locator format which must refer to a channel.

“Timer” then contains the timing information for the recording. The starttime should be given in seconds since epoch UTC, the duration in seconds. “repeatDays” is a bitmask specifying which days to repeat the recording on:

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| 0x01 | 0x02 | 0x04 | 0x08 | 0x10 | 0x20 | 0x40 |

If the value for “repeatDays” is prefixed with “0x”, it will be treated as a hexadecimal value. Otherwise, a decimal value is assumed.

The above message yields a simple response message:

```
<ltv:ProgramTimerResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>0</ltv:Result>
</ltv:ProgramTimerResponse>
```

As usual a result of “0” denotes success where a negative value is an error code. Note that since recording requests may be delayed an arbitrary time due to necessary user interaction in terms of conflict, “0” does not mean that the timer has necessarily been correctly programmed. It only means that the tv has accepted and processed the request.

## 9.14 Media search

The API also supports searching for media events and media items.

Use cases searching for media events include:

- EPG.
- Searching for PVR recordings by title
- Finding all youtube videos by a certain author
- Returning all mp3s on a USB stick that have been the first track on the original CD and have been created between 2003 and 2005.

Starting a search opens a logical session. Once the session has been opened, you can retrieve new result fragments from the search.

A search is started by use of a certain media directory (as identified by e.g. CHL0, FSL2, DLN1 etc.). After that, a set of constraining conditions is added to the search and a sorting order is defined. If available, you can fetch results from the search. Finally, the search session is closed.

### 9.14.1 Searching/sorting field selectors

To unify different search applications, some fields have specified meanings, regardless of the context of the application.

If applicable, these fields directly map the names of the corresponding item / entry properties.

| | | |
|---|---|---|
| Actor | E | One of the event's actor. |
| Album | E | |
| AlbumUUID | I | The uuid of the media item representing this entry's album. |
| AncestorUUID * | I | Media item uuid of ancestors.<br>This refers to the direct ancestor (i.e. parent) when filtering with EQUALS operator. This refers to some ancestor when filtered with LESSEQUAL. |
| Artist | E | The event's artist. |
| Caption * | I | Caption and shortCaption. |
| ExtendedInfo | E | Extended info, if available. |
| Genre | E | Theme / Genre of the media event. |
| MediaItemUUID * | I | |
| Mood | I | Mood associated with the station, if supported. |
| ShortInfo | E | Short info, if available. |
| StartTime * | E | The start time of event's natural availability. |
| StopTime | E | The end time of event's natural availability. |
| TrackNum | I | Number of track, if available. |
| User | I/E | Associated user information. |

Fields not listed in the table may be available, but do not have a standard meanings. Fields marked with an asterisk are mandantory to be supported in searches. "I" means that adding a restriction by use of the particular field discards media items from the result set, "E" means that it discards media events from the result set.

@@@ Insert subclass meanings here. @@@

### 9.14.2 Searching field operators

The following operators are defined to define constraints:

| | |
|---|---|
| EQUAL | An exact match of candidate and operand. |
| CONTAINS | The candidate contains the operand in a type specific way:<br>– Strings: True, if candidate contains the operand string.<br>– Pathes: True, if (normalized) candidate path contains (normalized) operand string literally.<br>– Numerical values: Not defined |
| LESS | The candidate is numerically or alphabetically smaller than the operand. |

| | |
|---|---|
| | – Strings and numerical values: Natural implementation.<br>– Pathes: Undefined operation. |
| LESSEQUAL | The candidate is numerically or alphabetically smaller or equal to the operand.<br>– Strings and numerical values: Natural implementation.<br>– Pathes: Undefined, behaves the same as equal. |
| GREATER | The candidate is numerically or alphabetically greater than the operand.<br>– Strings and numerical values: Natural implementation.<br>– Pathes: The candidate is some sub-directory (by the semantics of the media directory) of the operator. |
| GREATEREQUAL | The candidate is numerically or alphabetically greater or equal to the operand.<br>– Strings and numerical values: Natural implementation.<br>– Pathes: The candidate either is the same as the operand, or some sub-directory of the operator. |

@@@ XML example here @@@

### 9.14.3 Language considerations

Because string comparisons can depend on language specific collations, it is possible to add a language specification to a string compare restriction, both in selection and in sorting.

@@@ XML example here @@@

**Implementation note**

Supporting exactly one internationalization specification per search query is required. However, it is recommended to implement more.

### 9.14.4 Sorting the selected result set

The result set can be sorted by a list of selectors.

@@@ XML subtype example here @@@

### 9.14.5 Mapping for current SI info based EPG searches

**Time interval filter**

Today's time filter for EPG queries directly maps on the startTime property of the media event. An artificial endTime property is available to make time interval queries easier.

**Theme filter**

Today's theme property directly maps on the genre property of the media event.

**Filter by channel**

To filter by channel, use the media uuid filter and match it with the desired channel's uuid.

**Filter by service type**

As the service type (service indicating one of e.g. DVB-C, DVB-T, DVB-S, ATV etc. .) is represented by the particular subtype of the service, use a "is-a" filter on the item class.

### 9.14.6 UPnP mapping

As far as possible, the TV will map standard UPnP CLS browse/search calls to SOAP search requests. Please note, that providing meaningful information is not possible in all cases due to the limitations of the UPnP standard.

**Implementation Note**

The UPnP CDS implementation uses the same sources of information as the SOAP directories.

## 9.15 Get DR+ archives

Query all DR+ archive recordings, from internally or externally connected sources or network sources.

The results can be filtered by the parameters of element QueryParameters. The maximum number of items is limited to 1000. OrderField is not implemented yet, the results are ordered alphabetically and ascending.

```
<ltv:GetDRPlusArchive>
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1426235174</ltv:ClientId>
  <ltv:QueryParameters>
    <ltv:Range startIndex="0" maxItems="50"/>
    <ltv:OrderField field="userChannelNumber" type="ascending"/>
  </ltv:QueryParameters>
</ltv:GetDRPlusArchive>
```

The response provides various information related to recordings, if recordings have been found.

Element ResultItemDRPlusFragment is similar to  element ResultItemFragment. Attribute sequenceNumber can be used to sort several smaller fragments in the correct order. Attribute totalResults describes the number of results of all found DR+ recordings. Attribute returnedResults is the number of results with respect to QueryParameters attributes.

A single recording is described by element ResultItemDRPlus. Element MediaItemDRPlus describes the media item type, the UUID and the Locator, which can be used to start media playback.

Element MediaInformationDRPlus gives more information on the recording. If the media item is placed in a folder, the folder name is given by element Folder.

```
<ltv:GetDRPlusArchiveResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <ltv:fcid>?</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1426235174</ltv:ClientId>
  <ltv:ResultItemDRPlusFragment sequenceNumber="5023233" totalResults="2" returnedResults="2"
startIndex="0">
    <ltv:ResultItemDRPlus>
      <ltv:MediaItemDRPlus itemInfoClass="object.item.videoItem">
        <ltv:uuid>UUID_HDR_0f60601b-dc63-482a-acdd-521526134cdf_00000014</ltv:uuid>
        <ltv:Locator>drplus://UUID_HDR_0f60601b-dc63-482a-acdd-521526134cdf_00000014</ltv:Locator>
      </ltv:MediaItemDRPlus>
      <ltv:MediaInformationDRPlus>
        <ltv:Folder>QWERTY</ltv:Folder>
        <ltv:Title>ARD-Buffet</ltv:Title>
        <ltv:Subtitle>ARD-Buffet</ltv:Subtitle>
        <ltv:ServiceName>Das Erste HD</ltv:ServiceName>
        <ltv:LongDescription>Themen u.a.:
* Sören Anders bereitet Kabeljau auf Safranrisotto zu
* Zuschauerfragen zum Thema: Reisen mit Caravan und Wohnmobil; zu Gast  im Studio: Thomas Nitsch, ADAC;
außerdem: Thomas Niemietz, Couchsurfer in "Auf 3 Sofas durch ..."
* Allein durch Russland
* Kabeljau
* Auf 3 Sofas durch Mumbai
* Nadine Weckardt gestaltet eine Tischdeko in Apfelgrün mit Schneeball
* Quiz: "Ach was!"

Moderation: Evelin König
Produziert in HD</ltv:LongDescription>
        <ltv:Duration>00:08:03.000</ltv:Duration>
        <ltv:StartTime>12:40:57.000</ltv:StartTime>
        <ltv:EndTime>12:49:00.000</ltv:EndTime>
<ltv:StreamingUrl>http://192.168.10.107:1543/lt0/0/$0$UUID_HDR_0f60601b-dc63-482a-acdd-
521526134cdf_00000014</m:StreamingUrl>
<ltv:ImageUrl>http://192.168.10.107:1543/lt0/xetn/0/$0$UUID_HDR_0f60601b-dc63-482a-acdd-
521526134cdf_00000014</m:ImageUrl>
      </ltv:MediaInformationDRPlus>
    </ltv:ResultItemDRPlus>
    <ltv:ResultItemDRPlus>
      <ltv:MediaItemDRPlus itemInfoClass="object.item.videoItem">
        <ltv:uuid>UUID_HDR_0f60601b-dc63-482a-acdd-521526134cdf_00000015</ltv:uuid>
        <ltv:Locator>drplus://UUID_HDR_0f60601b-dc63-482a-acdd-521526134cdf_00000015</ltv:Locator>
     </ltv:MediaItemDRPlus>
        <ltv:MediaInformationDRPlus>
          <ltv:Folder>AX2</ltv:Folder>
          <ltv:Title>ARD-Mittagsmagazin</ltv:Title>
          <ltv:Subtitle>ARD-Mittagsmagazin</ltv:Subtitle>
          <ltv:ServiceName>Das Erste HD</ltv:ServiceName>
          <ltv:LongDescription>Themen u.a.:
* Das neue Kobane: Wie kurdische Kämpfer Tal Abiad vom IS zurückerobern wollen
* Europäischer Polizeikongress: Wie Fahnder dem "individuellen Dschihad" im Netz begegnen
* Ein Kind, drei Eltern: Wie Großbritannien mit "Designer-Babys" ein Tabu bricht

Moderation: Hannelore Fischer
Produziert in HD</ltv:LongDescription>
          <ltv:Duration>00:28:59.000</ltv:Duration>
          <ltv:StartTime>13:11:40.000</ltv:StartTime>
          <ltv:EndTime>13:40:39.000</ltv:EndTime>
<ltv:StreamingUrl>http://192.168.10.107:1543/lt0/0/$0$UUID_HDR_0f60601b-dc63-482a-acdd-
521526134cdf_00000014</m:StreamingUrl>
<ltv:ImageUrl>http://192.168.10.107:1543/lt0/xetn/0/$0$UUID_HDR_0f60601b-dc63-482a-acdd-
521526134cdf_00000014</m:ImageUrl>
        </ltv:MediaInformationDRPlus>
      </ltv:MediaInformationDRPlus>
    </ltv:ResultItemDRPlus>
  </ltv:ResultItemDRPlusFragment>
</ltv:GetDRPlusArchiveResponse>
```

## 9.16 Bridging JavaScript and SOAP

For some applications it is desirable to be able to exchange information between the JavaScript running in the currently opened page in the browser and and outside application via SOAP. Therefore it is possible to exchange key/value pairs between these two and keep track of changes.

**Current Limitations:**

At the time of this writing, up to 16 key/value pairs may be stored. Both key and value must be NUL-terminated strings. The key length must not exceed 64 byte including the NUL, the value length must not exceed 1024 byte including the NULL. Attempting to set an additional key will be ignored. The whole storage is reset when going back to the portal page or leaving the browser.

### 9.16.1 Sending a key/value pair to JavaScript

A key/value pair may be sent to JavaScript as follows:

```
<ltv:SetValueToBrowserJS>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Key>MyKey</ltv:Key>
  <ltv:Value>Some arbitrary value</ltv:Value>
</ltv:SetValueToBrowserJS>
```

This will always be confirmed with just the fcid and ClientId:

```
<ltv:SetValueToBrowserJSResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
</ltv:SetValueToBrowserJSResponse>
```

Note that this does not imply that the value was actually set in the JavaScript. If you want to make sure it was, you need to query it and compare.

### 9.16.2 Retrieving a value from JavaScript

A value for a given key may be queried as follows:

```
<ltv:GetValueFromBrowserJS>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Key>MyKey</ltv:Key>
</ltv:GetValueFromBrowserJS>
```

And will be answered i.e. like this:

```
<ltv:GetValueFromBrowserJSResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Key>MyKey</ltv:Key>
  <ltv:Value>Some arbitrary value</ltv:Value>
</ltv:GetValueFromBrowserJSResponse>
```

### 9.16.3 Retrieving the change list from JavaScript

In order to avoid unnecessary traffic, it is possible to get a list of keys that have been changed by JavaScript. A key will be marked as changed if its value has been changed by JavaScript. The mark will be reset when the value for that key is either read or set via SOAP.

```
<ltv:GetBrowserJSChangeList>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
</ltv:GetBrowserJSChangeList>
```

And will be answered i.e. like this:

```
<ltv:GetBrowserJSChangeListResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:ChangeList>MyKey,anotherKey</ltv:ChangeList>
</ltv:GetBrowserJSChangeListResponse>
```

### 9.16.4 Using the SOAP-Bridge from HTML / JavaScript

In order to communicate with the outside world via SOAP, one first needs to include the soapbridge object in your page, i.e.:

```
<object id="soapbridge" type="application/loewe-soapbridge" onValueChange="valueChange"/>
```

Then one may set values to it:

```
soapbridge = document.getElementById(“soapbridge”);
soapbridge.setValue( “anotherKey”, “some other value” );
```

Retrieve these values:

```
theValue = soapbridge.getValue( “MyKey” );
```

And register a callback to be notified of changes:

```
function valueChange( key, value )
{
   window.alert( "NOTIFY CHANGE OF " + key + " TO " + value );
}

soapbridge.onValueChange = valueChange;
```

Note that the “onValueChange” property in the object tag above has the same effect.

## 9.17 OSD message control

OSD message control allows requesting the display of a OSD message or action field with user-defined text.

The display time of this message can be set by the user (e.g. in menu System settings → Control → On-screen displays → Display time). The message can also be canceled by user interaction before the timeout.

The priority of this message in comparison to similar messages is set to medium, which is the most common priority for OSD messages. If multiple messages are requested to be displayed concurrently, priorities determine which request is processed first.

```
<ltv:SetActionField>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:InputText>Test</ltv:InputText>
</ltv:SetActionField>
```

gives a response

```
<ltv:SetActionFieldResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:SetActionFieldResponse>
```

and a OSD message with text “Test” gets displayed. If an empty InputText parameter is assigned, no OSD message gets displayed and the Result parameter of the response is ERR_EMPTY_STRING instead of OK.

## 9.18 Query supported features

Query supported features. Supported feature names same as loreg keys:

remote-app

remote-recording

dual-recording

ChannelSwitchTime

vtunersearch

DashboardDesign2

DvbRadioRecording

BrowseHistory

DrFolderMangement

MultiSelect

BluetoothBase

TvMagazineSorting

Multiroom-Syncplayback

Multiroom-RemoteServicelists

StationListMgmtForApps

```
<ltv:GetFeature>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Name>remote-app</ltv:Name>
</ltv:GetFeature>
```

gives a response

```
<ltv:GetFeatureResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Name>remote-app</ltv:Name>
  <ltv:Status>Enabled</ltv:Status>
</ltv:GetFeatureResponse>
```

Result parameter Status can either be Enabled or Disabled. A feature is disabled if its value is less than 1 or if the feature name does not exist.

## 9.19 Get and set TV settings

GetSettings gives various TV settings. Settings, which depend on disabled settings, e.g. “RemoterecEmailConfigurationType” depends on “RemoterecEnabled”, may be omitted.

SetSetting can be used to set a single setting. Supported Name values are: “MultiroomActive”, “WolEnable”. Supported Value values are: “1”, “0”.

```
<ltv:GetSettings>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
</ltv:GetSettings>
```

gives a response

```
<ltv:GetSettingsResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:MultiroomActive>1</ltv:MultiroomActive>
  <ltv:WolEnable>0</ltv:WolEnable>
  <ltv:RemoterecEnabled>1</ltv:RemoterecEnabled>
  <ltv:RemoterecEmailConfigurationType>0</ltv:RemoterecEmailConfigurationType>
  <ltv:RemoterecScanStartOfPeriod>0</ltv:RemoterecScanStartOfPeriod>
  <ltv:RemoterecScanEndOfPeriod>21600</ltv:RemoterecScanEndOfPeriod>
  <ltv:RemoterecEmailAddress>tv-xxxxxxxxxxxxxxx@loewe-dialogue.com</ltv:RemoterecEmailAddress>
  <ltv:RemoterecEmailSecurePin>1234</ltv:RemoterecEmailSecurePin>
  <ltv:StreamingShareServer>1</ltv:StreamingShareServer>
</ltv:GetSettingsResponse>
```

```
<ltv:SetSetting>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Name>WolEnable</ltv:Name>
  <ltv:Value>1</ltv:Value>
</ltv:SetSetting>
```

gives a response

```
<ltv:SetSettingResponse>
  <ltv:fcid>8138436</ltv:fcid>
  <ltv:ClientId>LRemoteClient-0-1314017969</ltv:ClientId>
  <ltv:Result>OK</ltv:Result>
</ltv:SetSettingResponse>
```

# 10 Technical notes

## 10.1 Media incarnation and lifecycle

A *media creator* creates the actual media item. A media creator is invisible in terms of system API and system design. Even if there technically are media creators in the system, media creators are intentionally not represented in this api.

As soon any *media directory* includes this item into a list, it becomes available in the system for the first time.

**Example: TV channel**
A TV channel becomes available in a channel list, if found during channel scan, or if announced in a service table etc.

**Example: mp3 file**
An mp3 file on a usb stick becomes available in the system, as soon the hotplug manager has discovered the stick, and the disk browse service can return it if queried to.

**Example: Internet radio**
An internet radio station becomes available as soon the listing service, e.g. an online directory service, returns it.

**Example: CD**
The tracks on a CD become available after the CD has been inserted, and after the CD has been recognized by the hotplug manager process.

A media item is not required to be physically present. It is even not required to physically or technically exist. At the end of the day, a media item is something that is playable on the TV.

A media item is abstract and quite useless until provided by a *media provider*. By providing a medium, it is associated with a series of *media events*, or one media event only.

It is the media provider's job to take *media information* (which does not carry any time information), and to associate it with some absolute time.

**Example: TV broadcaster (live)**
The program of a TV channel, let's say RTL, is some unreachable data on RTL's servers somewhere in one or several data centers. Or, it is something that happens live and doesn't exist yet. In any situation it is useless now, simply because you can't access it in any way.
But, even now, the TV channel is accessible as a media item listed in a directory service commonly known as channel list.
As soon RTL acts as media provider and starts playout of the data, it is associated with a sequence of media information (the title of the show, the production date, actors, etc). Media information is stored in a different database down at RTL's data center. (Simplified,) Media information associated with a certain extent in time and a media item becomes a media event. Upon reception in the TV, this media event is what we know as (in the DVB standard meaning) event info.
The TV acts as media player, displaying the media item's media content (the audio/video stream), and displaying the (current and anticipated) media events.

@@@

**Example: mp3 file**
A media item is stored somewhere on a USB stick. It carries media information that by definition of the mp3 format is describing this particular media item, as long the mp3 file is played back.
As soon you press play for this mp3, or when your playlist advances to this file, playout starts.
Now, the yet unscheduled media information becomes a media event.
Your media player displays the media event and plays back the media content. If you play back the media at a later point in time, it generates another media event with the same media information.

@@@

**Example: EPG information**

The EPG information user interface displays known media events.

@@@

**Example: PVR archive**
The PVR archive user interface displays the media information that would become a media event once "scheduling" the stored show.

@@@

**Example: Channel list**
The channel list displays usually media items that physically are TV channels of some kind. In some detail view it might display the currently and next valid media event if known.

@@@

## 11 Implementation Status

We distinguish 5 different implementation states:

- **FINAL –** The functionality is implemented as described and not likely to change.
- **BETA –** The functionality is implemented as described, yet may still be modified.
- **ALPHA** – This functionality is implemented for testing purposes only and may be removed or changed at any time.
- **NYI –** Not yet implemented. This functionality is for future use and not yet included.
- **OBS(OLETE)** – Not supported any more.

The status of the methods described is as follows at the time of this writing:

| **Version** | **PV7.5** | **PV8.1** | **PV8.11** | **PV8.21** | **SL2, >= PV1.10** |
|---|---|---|---|---|---|
| *Get/SetVolume* | **FINAL** | **FINAL** | **FINAL** | **FINAL** | **FINAL** |
| *Get/SetMute* | **NYI** | **NYI** | **NYI** | **FINAL** | **FINAL** |
| *GetChannelList* | **BETA** | **BETA** | **BETA** | **BETA** | **BETA** |
| *GetCurrentEvent / GetNextEvent* | **FINAL** | **FINAL** | **FINAL** | **FINAL** | **FINAL** |
| *GetCurrentPlayback* | **-** | **BETA** | **BETA** | **BETA** | **BETA** |
| *GetDeviceData* | **BETA** | **BETA** | **FINAL** | **FINAL** | **FINAL** |
| *GetMediaEvent* | **NYI** | **NYI** | **NYI** | **NYI** | **NYI** |
| *GetMediaItem* | **FINAL** | **FINAL** | **FINAL** | **FINAL** | **FINAL** |
| *InjectKeyboardKey* | **FINAL** | **FINAL** | **FINAL** | **FINAL** | **FINAL** |
| *InjectRCKey* | **BETA** | **BETA** | **FINAL** | **FINAL** | **FINAL** |
| *Wake on LAN* | **-** | **-** | **-** | **-** | **BETA** |
| *ProgramTimer* | **BETA** | **BETA** | **BETA** | **BETA** | **BETA** |
| *RequestAccess* | **BETA** | **BETA** | **FINAL** | **FINAL** | **FINAL** |
| *Subscribe* | **NYI** | **NYI** | **NYI** | **NYI** | **NYI** |
| *ZapToApplication* | **FINAL** | **FINAL** | **FINAL** | **FINAL** | **FINAL** |
| *ZapToMedia* | **FINAL** | **FINAL** | **FINAL** | **FINAL** | **FINAL** |
| (MediaSearch) | **NYI** | **NYI** | **NYI** | **NYI** | **NYI** |
| (PlayerControl) | **NYI** | **NYI** | **NYI** | **NYI** | **NYI** |