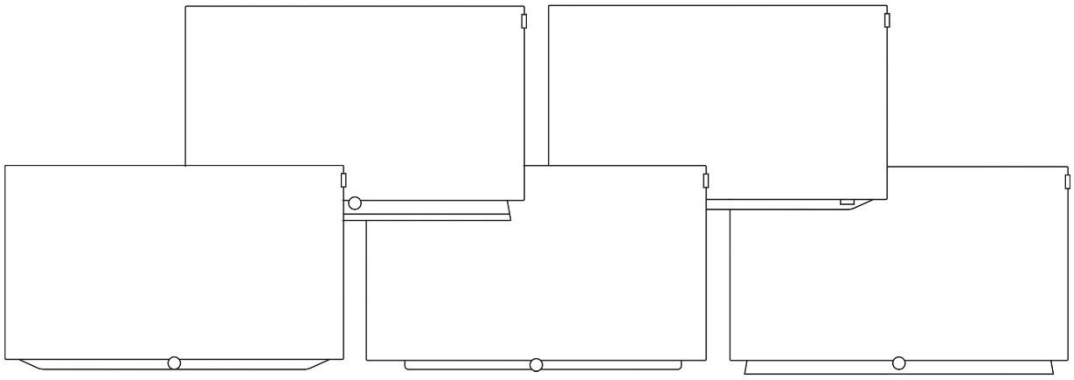
LOEWE TV
remote API



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.
1.0.43	12.02.16	TH		Modified GetCurrentPlayback, GetDRPlusArchive. Added GetVolumes, ParentalLock.
1.0.44	20.06.16	TH		Added GetCurrentStatus. Modified GetVolume, SetVolume, GetMute, SetMute. Modified GetSettings/SetSetting for UpnP Renderer.
1.0.45	27.10.16	TH		Added parameter WolInteractive to GetSettings/SetSetting. Added GetTimerList.
1.0.46	07.11.17	TH		- Changed description of QueryParameters for GetChannelList. - Changed description of Wake on LAN. - Added user defined max volume for SetVolume. - Added list ID to media item locator, to be able to tune to service in specific list. - Added channel number to GetTimerList. - Added timeout flag to SetActionField. - Added GetActionField. - Changed ZapToMedia and GetCurrentPlayback to be able to query and change PIP playback. - Added GetParentalLock and added locked status to GetCurrentStatus. - Added CreateIterator and CloseIterator. - Added audio/language, subtitle selection and network host name to GetSettings/SetSetting. - Added support for file URL to ZapToMedia, to be able to play video connected via USB. - Added thumbnail URL for the HDMI channels to GetMediaItem and GetChannelList. - Added resume position to response of GetDRPlusArchive. - Added support for many Unicode characters to InjectKeyboardKey. - Added ActivateFeaturePackage for activating an upgrade feature package by article number. Updated list of supported features. - Removed or replaced comments. Removed description of media player control. Rephrased some texts.

1.0.47	07.08.18	TH		- Added Feature Upgrade FullCharacterKeyboard. - Added DeactivateFeaturePackage for deactivating a Feature Upgrade. - Added PENDING result to responses for ActivateFeaturePackage and DeactivateFeaturePackage. - When a recording timer is programmed, its default title is replaced if it can be matched to an EPG event. - Global setting whether to record subtitles or not is honored. - Added settings.txt to export and import of service lists. - Removed status "undefined" for GetCurrentPlayback. - Added hotel remote control key codes.

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
	UPnP directory service specification.
	id3v2 information

2.2 Informative References

Document	Contents

2.3 Related LOEWE specifications

Document	Contents

3 Open Issues

4 Table of Contents

1 Revision History 2
2 Related Documents 5
2.1 Normative references 5
2.2 Informative References 5
2.3 Related LOEWE specifications 5
3 Open Issues 5
4 Table of Contents 6
5 Introduction 9
5.1 Abstract 9
5.2 Portability and requirements 9
6 Protocol 9
6.1 Transport level 9
6.2 SOAP content 9
6.3 Reused from UPnP 10
7 Concepts 11
7.1 SOAP specifics 11
7.1.1 Subscription manager 11
7.1.2 Events 11
7.1.3 SOAP over UDP 11
7.2 Media handling 11
7.2.1 Media items 11
7.2.2 Media information 11
7.2.3 Media event 11
7.3 TV operating modes 13
7.4 Internationalization 13
7.5 List queries 13
8 Types 14
8.1 Subscriber 14
8.2 Notification 14
8.3 Input events 14
8.4 Input event targets 14
8.4.1 Legacy RC key events 14
8.4.2 Legacy special purpose input events 14
8.4.3 Unbound keyboard input 14
8.4.4 Targeted keyboard input 14
8.5 Media directory 15
8.5.1 Media directory purposes 15
8.6 Media creator 16
8.7 Media item 16
8.7.1 Media Item uuid 18
8.8 Media information 18
8.9 Media provider 19
8.10 Media event 19
8.11 Event sources 19
8.12 Result item list fragment 20
8.13 Result event list fragment 20
9 Open API methods 21
9.1 Connecting 21
9.1.1 Pairing 21
9.2 Retrieving Device Information 24
9.3 Subscribing 24

9.3.1 Subscribe 24
9.4 Injecting 25
9.4.1 Inject RC event 26
9.4.2 Inject Keyboard event 28
9.4.3 Wake On LAN 30
9.4.4 Local control 30
9.4.5 Volume control 31
9.4.6 Mute control 31
9.5 About media items 32
9.5.1 Get Media Items 32
9.6 About media events 34
9.6.1 Get media events 34
9.7 Channel lists 35
9.7.1 Get channel list 35
9.7.2 Get list of channel lists 36
9.7.3 Add personal channel list 38
9.7.4 Remove personal channel list 38
9.7.5 Rename personal channel list 39
9.7.6 Add services to a personal channel list 39
9.7.7 Remove services from a personal channel list 40
9.7.8 Move services in a personal channel list 40
9.7.9 Import service lists 41
9.7.10 Export service lists 43
9.7.11 Get list of virtual lists 44
9.7.12 Add personal channel lists from virtual 46
9.8 Channel information 47
9.8.1 Get channel info 47
9.9 Changing channels 48
9.9.1 Remote control like zapping 48
9.9.2 Switching to another application 48
9.10 Current playback information 49
9.11 Current status information 50
9.12 Event information 52
9.12.1 Get current event 52
9.12.2 Get next event 52
9.13 Programming Timers 54
9.13.1 Getting the list of timers 55
9.14 Media search 57
9.14.1 Searching/sorting field selectors 57
9.14.2 Searching field operators 58
9.14.3 Language considerations 58
9.14.4 Sorting the selected result set 59
9.14.5 Mapping for current SI info based EPG searches 59
9.14.6 UPnP mapping 59
9.15 Get DR+ archives 59
9.16 Bridging JavaScript and SOAP 61
9.16.1 Sending a key/value pair to JavaScript 61
9.16.2 Retrieving a value from JavaScript 62
9.16.3 Retrieving the change list from JavaScript 63
9.16.4 Using the SOAP-Bridge from HTML / JavaScript 63
9.17 OSD message control 63
9.17.1 Set OSD message 63
9.17.2 Get OSD message 64

9.18 Feature Upgrades 65
9.18.1 Query supported features 65
9.18.2 Activate feature package 67
9.18.3 Deactivate feature package 67
9.19 Get and set TV settings 67
9.20 Query (DR+ archive) volumes 69
9.21 Parental Lock control 70
9.21.1 Set parental lock setting 70
9.21.2 Get parental lock setting 72
9.22 Iterators 72
9.22.1 Create iterator 72
9.22.2 Close iterator 74
10 Technical notes 75
10.1 Media incarnation and lifecycle 75
11 Implementation Status 77

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.

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 whenever possible. Expect long latency times from the TV.

5.2 Portability and requirements

To ensure compatibility industry standard protocols are used whenever 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.

The LOEWE TV provides resource loewe_tablet_0001 for communication via HTTP, so the header has to include a request line and a header field similar to this:

```
POST /loewe_tablet_0001 HTTP/1.1
Host: ip-address:905
```

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

WS eventing is only partially supported. UPnP notifications are used sometimes as an alternative.

7.1.3 SOAP over UDP

SOAP over UDP is not supported.

7.2 Media handling

This is a short practical overview for a more in-depth discussion of media data types refer to chapter **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, I 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 info
- An 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.

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 information, 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").

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 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

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

8.4.2 Legacy special purpose input events

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

...

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

8.4.3 Unbound keyboard input

Unbound keyboard input are input events that are not targeted to a particular focusable 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

Targeted keyboard input is not supported.

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 refers to something connected via (IP based) network.
PURPOSE_VIA_LOGICAL_FS	This volume refers 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 time shifting.
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.

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.

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

```
<m:MediaInformation>
 <m:ShortInfo>The River</m:Caption>
 <m:ExtendedInfo m:lang="en">
  A journey into the depth of the Amazonas, exploring ...
 </m:ExtendedInfo>
 <m:ExtendedInfo m:lang="de">
  Eine Reise in die Welten des Amazonas. Entdecken Sie ...
 </m:ExtendedInfo>
</m: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. 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 information. Colon or comma-separated list of genres.

Media information items can carry more informational items.

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.

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

Description:

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

8.11 Event sources

Events can be generated by event sources. Typical event sources include an 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

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

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

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

might yield the following response:

```
<m:RequestAccessResponse>
 <m:fcid>8138941</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:AccessStatus>Pending</m:AccessStatus>
</m: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 re-identify:

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

And might yield the following response:

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

AccessStatus can either “Denied”, “Pending” or “Accepted”.

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 it 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.

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

might yield the following response:

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

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

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 interfere 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 behavior or successful operation due to the requirement's nature.

9.4.1 Inject RC event

To emulate legacy behavior, 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 behavior, 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.

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

In order to properly simulate a long key-press 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	TIMER	91

Key	Value	Key	Value
MENU	11	DR_ARCHIVE	92
TV_OFF	25	AV1	114
GREEN	26	AV2	115
RED	27	AV3	116
UP	32	AVS	117
DOWN	33	VGA	118
PIC	35	HDMI1	119
OK	38	COMP	120
BLUE	40	HDMI2	121
YELLOW	43	HDMI3	122
MEDIA	49	HDMI4	123
RADIO	53	VIDEO	124
TTX	60	SPDIF_IN	125
END	63	HOME (HOTEL)	126
SOUND	64	RESET (HOTEL)	127
BACK	65	PICTURE (HOTEL)	128
TV_MODE_ON	72	ALARM (HOTEL)	129
RADIO_MODE_ON	73	SLEEP (HOTEL)	130
EPG_STAR	78	HOTEL (HOTEL)	131
INFO	79	PICTURE FORMAT (HOTEL)	132
PIP_HASH	88	SOUND (HOTEL)	133
ASPECT	90		

9.4.1.2 Key codes for alphabet I2700-hdr

Alphabet “I2700-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 acknowledged, and successful delivery cannot be granted.

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

The *mode* attribute is just the same as for RC keys, i.e. one of (press, release, repeat). The *modifiers* are a comma separated 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

The settings of Wake on LAN/WLAN can be queried and set using “GetSettings” and “SetSetting” (Get and set TV settings, chapter 9.20).

After the TV has been switched on by Wake on LAN/WLAN magic packet it either is in network standby or interactive mode, which is controlled by setting “WolInteractive”.

In network standby the TV can be switched on to interactive mode by sending RC key code TV_ON/22 (Inject RC event, chapter 9.41):

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

9.4.4 Local control

See InjectRCKey.

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).

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

The *value* is in range 0 – 999999. Currently the API discards the last 4 decimal places.

It is also possible to ask for the current volume level:

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

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

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

```
<m:GetMute>
   <fcid>18725463</fcid>
   <ClientId>LRemoteClient-0-1314017969</ClientId>
</m: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 call retrieves a standard set of information about a single media item. 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>chl0:00-00080001ffff03fb283d</m:uuid>
       <m:AncestorUuid>sl13eb69b6-5331-420a-ade2-630b11d3cb83</m:AncestorUuid>
       <m:Locator>channel://1:ch5098e8a4-3194-415c-afc3-7a1727982395:sl13eb69b6-5331-420a-ade2-
630b11d3cb83</m:Locator>
       <m:Caption>1</m:Caption>
     </m:MediaItem>
     <m:MediaInformation>
       <m:ThumbnailURL>http://172.21.3.141:1543/lt0/xetn/0/$1$sl13eb69b6-5331-420a-ade2-
630b11d3cb83$ch5098e8a4-3194-415c-afc3-7a1727982395</m:ThumbnailURL>
       <m:ShortInfo>Das Erste HD</m:ShortInfo>
       <m:Attributes>76</m:Attributes>
       <m:StreamingUrl>http://172.21.3.141:1543/lt0/0/$1$sl13eb69b6-5331-420a-ade2-
630b11d3cb83$ch5098e8a4-3194-415c-afc3-7a1727982395</m:StreamingUrl>
       <m:OriginalStreamingUrl/>
     </m:MediaInformation>
   </m:ResultItem>
</m:GetMediaItemResponse>
```

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.

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

might yield the following response:

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

```
<urn:GetChannelList>
  <fcid>?</fcid>
  <ClientId>LRemoteClient-0-1510040845</ClientId>
  <ChannelListView>fastscansl13eb69b6-5331-420a-ade2-630b11d3cb83</ChannelListView>
  <QueryParameters>
    <Range startIndex="0" maxItems="3"/>
    <MediaItemInformation>true</MediaItemInformation>
    <MediaItemClass></MediaItemClass>
  </QueryParameters>
</urn:GetChannelList>
```

ChannelListView		View name obtained from GetListOChannelLists	
QueryParameters	Range	Start index and maximum number of items returned	
QueryParameters	MediaItemInformation	Add media item information to the response.	true/false
QueryParameters	MediaItemClass	Filter only for TV or radio channels.	video/audio/undefined/[empty]

```
<m:GetChannelListResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1510040845</m:ClientId>
  <m:ChannelListView>fastscansl13eb69b6-5331-420a-ade2-630b11d3cb83</m:ChannelListView>
  <m:ChannelListName>ASTRA1 19,2°E</m:ChannelListName>
  <m:AncestorUuid>sl13eb69b6-5331-420a-ade2-630b11d3cb83</m:AncestorUuid>
  <m:ItemClass>undefined</m:ItemClass>
  <m:ResultItemFragment sequenceNumber="9076678" totalResults="1218" returnedResults="3"
startIndex="0" hash="4294967295">
    <m:ResultItemReference mediaItemUuid="chl0:00-00080001ffff03fb283d" itemClass="video"
shortInfo="Das Erste HD" locator="channel://1:ch5098e8a4-3194-415c-afc3-7a1727982395:sl13eb69b6-5331-
420a-ade2-630b11d3cb83" caption="1" thumbnailUrl="http://172.21.3.141:1543/lt0/xetn/0/$1$sl13eb69b6-
5331-420a-ade2-630b11d3cb83$ch5098e8a4-3194-415c-afc3-7a1727982395"
streamingUrl="http://172.21.3.141:1543/lt0/0/$1$sl13eb69b6-5331-420a-ade2-630b11d3cb83$ch5098e8a4-
3194-415c-afc3-7a1727982395" attributes="76" originalStreamingUrl=""/>
    <m:ResultItemReference mediaItemUuid="chl0:00-00080001ffff03f32b66" itemClass="video"
shortInfo="ZDF HD" locator="channel://2:chf5a4669e-dec4-4881-b771-5f2245d9458a:sl13eb69b6-5331-420a-
ade2-630b11d3cb83" caption="2" thumbnailUrl="http://172.21.3.141:1543/lt0/xetn/0/$1$sl13eb69b6-5331-
420a-ade2-630b11d3cb83$chf5a4669e-dec4-4881-b771-5f2245d9458a"
streamingUrl="http://172.21.3.141:1543/lt0/0/$1$sl13eb69b6-5331-420a-ade2-630b11d3cb83$chf5a4669e-
dec4-4881-b771-5f2245d9458a" attributes="76" originalStreamingUrl=""/>
    <m:ResultItemReference mediaItemUuid="chl0:00-00040001ffff04412ee3" itemClass="video"
shortInfo="RTL Television" locator="channel://3:ch0f58f899-4add-4a94-a753-2bad1e655252:sl13eb69b6-
5331-420a-ade2-630b11d3cb83" caption="3"
thumbnailUrl="http://172.21.3.141:1543/lt0/xetn/0/$1$sl13eb69b6-5331-420a-ade2-
630b11d3cb83$ch0f58f899-4add-4a94-a753-2bad1e655252"
streamingUrl="http://172.21.3.141:1543/lt0/0/$1$sl13eb69b6-5331-420a-ade2-630b11d3cb83$ch0f58f899-
4add-4a94-a753-2bad1e655252" attributes="72" originalStreamingUrl=""/>
  </m:ResultItemFragment>
</m:GetChannelListResponse>
```

ChannelListView		View name obtained from the request.	
ChannelListName		Actual name of the list.	
AncestorUuid		ID of this list.	
ItemClass		Item class filter from the request.	

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.

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

The response to the query might look like the following.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
<m:ImportServiceLists>
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1426235174</m:ClientId>
  <m: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>
  </m:Data>
</m: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.

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

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

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

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

9.8 Channel information

9.8.1 Get channel info

(deprecated, please use GetMediaItem)

9.9 Changing channels

9.9.1 Remote control like zapping

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

```
<m:ZapToMedia>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Player>0</m:Player>
 <m:Locator>channel://1:ch5098e8a4-3194-415c-afc3-7a1727982395:sl13eb69b6-5331-420a-ade2-
630b11d3cb83</m:Locator>
</m:ZapToMedia>
```

Player	Player ID for main and PIP player	"0" (main), "1" (PIP)
Locator	TV/Radio channel, recording or media file	

The locator can be for a TV/Radio channel ("channel://1:ch5098e8a4-3194-415c-afc3-7a1727982395:sl13eb69b6-5331-420a-ade2-630b11d3cb83" from GetChannelList/GetMediaItem/CreateIterator), recording ("drplus://UUID_HDR_7d1d9329-c7d9-4323-87cb-e86420445eb7_00000013" from GetDRPlusArchive/CreateIterator) or media file ("file:///mnt/lo_005/video.mp4" from CreateIterator).

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

The "Result" field is an indicator whether the switching was successful. 0 denotes success. The response might contain additional information.

9.9.2 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.

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

will most likely generate a response like:

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

9.10 Current playback information

Information about what is currently shown on the TV screen.

```
<m:GetCurrentPlayback>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Player>0</m:Player>
</m:GetCurrentPlayback>
```

Player	Player ID of main and PIP player.	“0” (main), “1” (PIP)

```
<m:GetCurrentPlaybackResponse>
 <m:fcid>8138947</ltv.fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Mode>tv</m:Mode>
 <m:Locator>channel://6:ch210f59d2-d505-42dd-96e7-e299c6279835:y</m:Locator>
 <m:MediaItemUuid>chl0:00-00040001ffff04412f1c</m:MediaItemUuid>
</m:GetCurrentPlaybackResponse>
```

Mode	General mode of the TV.	“tv”, “radio”, “drplus”, “browser”, “UNKNOWN”
Locator	URI of main channel if mode is “tv”.	
MediaItemUuid	UUID of main media item if mode is “tv”.	

9.11 Current status information

Information about the current state of the TV.

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

```
<m:GetCurrentStatusResponse>
 <m:fcid>8138947</ltv.fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Power>tv</m:Power>
 <m:HdrPlayerState>playback</m:HdrPlayerState>
 <m:HdrSpeed>play</m:HdrSpeed>
 <m:SystemLocked>unlocked</m:SystemLocked>
</m:GetCurrentPlaybackResponse>
```

Power	Power status.	“idle”, “tv”
HdrPlayerState	General DR+/HDR player status.	“idle”, “playback”, “timeshift_playback”, “media_prepare”
HdrSpeed	DR+/HDR playback speed status.	“pause”, “forw_fast_max”, “forw_fast_high”, “forw_fast_medium”, “play”, “forw_slow_max”, “forw_slow_high”, “forw_slow_medium”, “forw_slow”, “back_fast_max”, “back_fast_high”, “back_fast_medium”, “back”, “back_slow_max”, “back_slow_high”, “back_slow_medium”, “back_slow”
SystemLocked	System lock status set by parental lock.	“unlocked”, “locked”

For some use cases, it makes sense to evaluate elements in combination. For example, a paused DR+ recording playback is specified by HdrPlayerState “playback” and HdrSpeed “pause”. HdrSpeed has only meaning when HdrPlayerState values “playback” and “timeshift_playback” are set.

Power:

- “tv”: Device is in interactive mode, accepts control commands.
- “idle”: Device is in standby mode, accepts only some control commands.

HdrPlayerState:

- “idle”: Player is not active.
- “playback”: Playback is active.
- “timeshift_playback”: Playback active of a file, which is currently recording.
- “media_prepare”: Playback is being prepared.

HdrSpeed:

- “pause”: Speed of playback is 0.
- “play”: Normal playback speed.
- Other values: playback trickmode.

9.12 Event information

9.12.1 Get current event

The following query

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

would yield the following response

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

```
<urn:GetNextEvent>
  <fcid>?</fcid>
  <ClientId>LRemoteClient-0-1510040845</ClientId>
  <Player>0</Player>
</urn:GetNextEvent>
```

would yield the following response

```
<m:GetNextEventResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ItemLocator>channel://1:ch5098e8a4-3194-415c-afc3-7a1727982395</m:ItemLocator>
  <m:Locator>channel://1:ch5098e8a4-3194-415c-afc3-7a1727982395</m:Locator>
  <m:LiveEvent>
    <m:Name>Wer weiß denn sowas?</m:Name>
    <m:ExtendedInfo>Some text.</m:ExtendedInfo>
    <m:Availability>
      <m:ScheduledTime startTime="1510049700" duration="2700"/>
    </m:Availability>
  </m:LiveEvent>
</m:GetNextEventResponse>
```

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:

```
<m:ProgramTimer>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Device>PVR</m:Device>
 <m:RecordingType>default</m:RecordingType>
 <m:Locator>
dvb://localhost/#?chlview=default&amp;progNum=2&amp;gcn=7&amp;onid=1&amp;tsid=1011&amp;sid=1111
0
 </m:Locator>
 <m:Timer starttime="1315494000" duration="3600" repeatDays="0"/>
</m: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.

If the requested timer programming can be matched successfully to an EPG event, the default timer title "Recording" is replaced by the title of EPG event.

The global setting whether to record subtitles or not is honored.

The above message yields a simple response message:

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

9.13.1 Getting the list of timers

Get list of currently active timer entries.

Example:

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

```
<m:GetGetTimerListResponse>
 <m:fcid>8138947</ltv.fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:ResultTimerListFragment totalResults="1">
  <m:ResultTimerList>
   <m:EntryUuid>1</m:EntryUuid>
   <m:EntryTitle>Auf Streife - Die Spezialisten</m:EntryTitle>
   <m:SourceUuid>ch86ca1be7-3cbd-49e2-85e7-e2183625de9c</m:SourceUuid>
   <m:DestinationId>0</m:DestinationId>
   <m:Start>1477555259</m:Start>
   <m:End>1477558732</m:End>
   <m:DescramblingAt>0</m:DescramblingAt>
   <m:RecordingType>1</m:RecordingType>
   <m:Crid><![CDATA[<?xml version="1.0" encoding="utf-8"?>
<CridsData
Version="1"><Crids/><Children/><RequestedCrid>0</RequestedCrid><CheckAuthority>false</CheckAutho
rity><HasRecommendations>false</HasRecommendations><IsRecommendation>false</IsRecommendation
><IsSerial>false</IsSerial></CridsData>]]></m:Crid>
   <m:LockPin>0</m:LockPin>
   <m:ProgPin>0</m:ProgPin>
   <m:Attributes>4</m:Attributes>
   <m:SourceAncestorUuid>sI59e1e3c0-eb89-4ee9-8758-076874e78360</m:SourceAncestorUuid>

<m:SourceUri><![CDATA[rec://plain/dvbs?frontend=4&satid=4&coderate=9&frequency=12545023&symbolr
ate=22000&inversion=1&modulation=2&polarization=2&band=2&onid=1&sid=17500&tsid=1107]]></m:Sourc
eUri>
   <m:SourceName>SAT.1</m:SourceName>
   <m:SourceLocator/>
   <m:DestinationVolumeUri/>
   <m:DestinationVolumeId/>
   <m:EpgEventId>105630</m:EpgEventId>
   <m:SourceLocatorName/>
   <m:DestinationVolumeIdName/>
   <m:SeriesTag/>
   <m:NumColumns/>
  </m:ResultTimerList>
 </m:ResultTimerListFragment>
</m:GetTimerListResponse>
```

EntryUuid	ID used delete or edit existing timer entries.	
EntryTitle		
SourceUuid	Source channel UUID.	
DestinationId	Destination recorder ID.	
Start	Expected start time in UTC epoch time [seconds].	

End	Expected end time in UTC epoch time [seconds].	
DescramblingAt	Descramble time in UTC epoch time [seconds].	
RecordingType	Type of recording bitmask.	0x0 (Undefined), 0x1 (Once), 0x2 (Monday till Friday), 0x4 (Daily), 0x8 (Weekly), 0x10 (Serial), 0x20 (User notification/Memo), 0x40 (Recode/Descramble), 0x80 (Copy), 0x100 (Multiroom playback)
Crid	Content Reference Identifier.	
LockPin	PIN for child lock.	
ProgPin	CI/CI+ CAM PIN.	
Attributes	Attributes of recording bitmask.	0x1 (Delete protection), 0x2 (Auto time control), 0x4 (Subtitles), 0x8 (Descramble), 0x10 (Descramble overnight), 0x20 (Conflict), 0x40 (Add to dashboard), 0x80 (Recording active), 0x200 (No overlap), 0x400 (Moveable in conflict), 0x800 (Is recommendation), 0x1000 (Force no overlap), 0x2000 (Is part of series)
SourceAnchestorUuid	Source channel list ID.	
SourceUri	Source channel list URI.	
SourceName	Source channel list name.	
SourceLocator	Item locator, e.g. used for copy	
DestinationVolumeUri	Not supported yet.	
DestinationVolumeId	ID of target volume, e.g. used for copy.	
EpgEventId	UUID of an EPG Event.	
SourceLocatorName	Media item locator friendly name.	
DestinationVolumeIdName	Volume friendly name.	
SeriesTag	Tag to define a belonging to some series of recordings.	
NumColumns	For internal use.	

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. 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 mandatory 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.

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: – Strings: True, if candidate contains the operand string. – Pathes: True, if (normalized) candidate path contains (normalized) operand string literally. – Numerical values: Not defined
LESS	The candidate is numerically or alphabetically smaller than the operand. – Strings and numerical values: Natural implementation. – Pathes: Undefined operation.
LESSEQUAL	The candidate is numerically or alphabetically smaller or equal to the operand. – Strings and numerical values: Natural implementation. – Pathes: Undefined, behaves the same as equal.
GREATER	The candidate is numerically or alphabetically greater than the operand. – Strings and numerical values: Natural implementation. – 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. – Strings and numerical values: Natural implementation. – Pathes: The candidate either is the same as the operand, or some sub-directory of the operator.

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.

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.

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.

```
<m:GetDRPlusArchive>
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1426235174</m:ClientId>
  <m:QueryParameters>
    <m:Range startIndex="0" maxItems="1"/>
    <m:OrderField field="userChannelNumber" type="ascending"/>
  </m:QueryParameters>
  <m:VolumeId>FSL2://8549-5403</m:Volume>
</m:GetDRPlusArchive>
```

Range	startIndex	Skip results before this index.	Unsigned integer
	maxItem	Limit results.	Maximum 1000
OrderField		Not implemented yet. Results ordered alphabetically and ascending.	
VolumeId		Optional. Get results for a specific volume. The unique volume ID can be obtained from call GetVolumes. If not set, the default DR+ archive of the integrated hard disk drive or external memory is queried for the results.	

```
<m:GetDRPlusArchiveResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1426235174</m:ClientId>
  <m:ResultItemDRPlusFragment sequenceNumber="5023233" totalResults="104" returnedResults="1"
startIndex="0">
    <m:ResultItemDRPlus>
      <m:MediaItemDRPlus itemInfoClass="object.item.videoItem">
        <m:uuid>UUID_HDR_7d1d9329-c7d9-4323-87cb-e86420445eb7_00000116</m:uuid>
        <m:Locator>drplus://UUID_HDR_7d1d9329-c7d9-4323-87cb-e86420445eb7_00000116</m:Locator>
      </m:MediaItemDRPlus>
      <m:MediaInformationDRPlus>
        <m:Folder/>
        <m:Title>SOKO Wismar</m:Title>
        <m:Subtitle>Laura</m:Subtitle>
        <m:ServiceName>ZDF HD</m:ServiceName>
        <m:LongDescription>Some text.</m:LongDescription>
        <m:Duration>00:00:48.000</m:Duration>
        <m:StartTime>2017-09-29T11:51:09+01:00</m:StartTime>
        <m:EndTime>2017-09-29T11:51:57+01:00</m:EndTime>
        <m:StreamingUrl>http://172.21.3.141:1543/lt0/0/$0$UUID_HDR_7d1d9329-c7d9-4323-87cb-
e86420445eb7_00000116</m:StreamingUrl>
        <m:ImageUrl>http://172.21.3.141:1543/lt0/xetn/0/$0$UUID_HDR_7d1d9329-c7d9-4323-87cb-
e86420445eb7_00000116</m:ImageUrl>
        <m:Attributes>8193</m:Attributes>
        <m:ResumePos>27773</m:ResumePos>
      </m:MediaInformationDRPlus>
    </m:ResultItemDRPlus>
  </m:ResultItemDRPlusFragment>
</m:GetDRPlusArchiveResponse>
```

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

ResultItemDRPlusFragment		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	
ResultItemDRPlusFragment	ResultItemDRPlus	Single recording.	
ResultItemDRPlus	MediaItemDRPlus	Media item type, ID and locator, to be able to start playback.	
ResultItemDRPlus	MediaInformationDRPlus	Detailed information about the recording.	
MediaInformationDRPlus	Title	Program title.	
MediaInformationDRPlus	Subtitle	Program sub title.	
MediaInformationDRPlus	ServiceName	Name of the channel.	
MediaInformationDRPlus	LongDescription	Extensive description of the program.	

MediaInformationDRPlus	Duration	Duration of the recording.	
MediaInformationDRPlus	StartTime	ISO 8601 with time zone designator.	
MediaInformationDRPlus	EndTime	ISO 8601 with time zone designator.	
MediaInformationDRPlus	StreamingUrl	Streaming URL for the recording.	
MediaInformationDRPlus	ImageUrl	URL for the preview image for the recording.	
MediaInformationDRPlus	Attributes	Various attributes of the recording.	
MediaInformationDRPlus	ResumePos	The current playback position in milliseconds from the beginning of the recorded program.	

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:

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

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

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

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

And will be answered i.e. like this:

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

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

And will be answered i.e. like this:

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

9.17.1 Set OSD message

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

The duration 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. A message can either be created with or without the user defined timeout, by setting flag "IsTimeout" to "1" or "0".

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.

```
<m:SetActionField>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:InputText>Test</m:InputText>
 <m:IsTimeout>0</m:isTimeout>
 <m:Selectors>
  <m:Selector>OK</m:Selector>
  <m:Selector>Escape</m:Selector>
 </m:Selectors>
</m:SetActionField>
```

gives a response

```
<m:SetActionFieldResponse>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Result>OK</m:Result>
 <m:RequestId>47835</m:RequestId>
</m:SetActionFieldResponse>
```

and a OSD message with text “Test”, no timeout and button OK gets displayed. This message can either be confirmed by OK key press or escaped from by END/BACK key press.

The RequestId parameter of the response can be used to query the message status via GetActionField. The RequestId is not persistent over a restart.

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 and RequestId is “-1”.

Valid values for element Selector are:

OK	“Ok” button, to confirm/consume this message positively
Cancel	“Cancel” button, to confirm/consume the message negatively
Yes	“yes” button, to confirm/consume this message positively
No	“no” button, to confirm/consume the message negatively
Retry	“Retry” button, to confirm/consume the message with retry flag
Ignore	“Ignore” button, to confirm/consume the message with ignore flag
Escape	Invisible, allows escaping by END/BACK key press

9.17.2 Get OSD message

Query status of an OSD message previously set by SetActionField.

Example:

```
<m:GetActionField>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:RequestId>47835</m:RequestId>
</m:GetActionField>
```

gives a response

```
<m:GetActionFieldResponse>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Status>ConsumedPositive</m:Status>
</m:GetActionFieldResponse>
```

Valid values for Status are:

Unconfirmed	Message not confimed yet
Error	Some kind of error happened
Rejected	Message was rejected
Replaced	Message was replaced by another message
ConsumedPositive	Message was consumed/confirmed positively, e.g. by “Ok” or “yes”
ConsumedNegative	Message was consumed/confirmed negatively, e.g. by “no”
ConsumedRetry	Message was consumed/confirmed by “Replace”
ConsumedIgnore	Message was consumed/confirmed by “Ignore”
ConsumedEscape	Message was escaped from by BACK/END key press
ConsumedText	Not supported
ConsumedPassword	Not supported
ConsumedClose	Not supported
CamAtOnce	Not supported
CamAfterSwitchOff	Not supported
Timeout	Message was closed by user defined timeout
Info	Not supported
Unsupported	Not supported

9.18 Feature Upgrades

9.18.1 Query supported features

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

remote-app
remote-recording
dual-recording
ChannelSwitchTime
vtunersearch

DashboardDesign2
DvbRadioRecording
BrowseHistory
DrFolderMangement
MultiSelect
BluetoothBase
TvMagazineSorting
Multiroom-Syncplayback
Multiroom-RemoteServicelists
StationListMgmtForApps
SkipTimeshiftHint
EpgKeywordSearch
DashboardFolders
ArtistRecording
ExtendedVidRecExport
DrNetworkStorage
ScreenCast
GracenotePremiumServices
CompressedRecordings
FullCharacterKeyboard

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

gives a response

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

9.18.2 Activate feature package

Activate a feature upgrade package using the article number. Only certain packages can be activated this way (“bulk activation method” for codes not tied to a specific device).

```
<urn:ActivateFeaturePackage>
  <fcid>?</fcid>
  <ClientId>LRemoteClient-0-1510040845</ClientId>
  <Package>Some article number</Package>
</urn:ActivateFeaturePackage>
```

gives a response

```
<m:ActivateFeaturePackageResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1510040845</m:ClientId>
  <m:Package>Some article number</m:Package>
  <m:Result>ERROR</m:Result>
</m:ActivateFeaturePackageResponse>
```

Result can either be “ERROR”, “SUCCESS” or “PENDING.

9.18.3 Deactivate feature package

Deactivate a feature upgrade package using the article number. Only certain packages can be deactivated this way.

```
<urn:DeactivateFeaturePackage>
  <fcid>?</fcid>
  <ClientId>LRemoteClient-0-1510040845</ClientId>
  <Package>Some article number</Package>
</urn:DeactivateFeaturePackage>
```

gives a response

```
<m:DeactivateFeaturePackageResponse xmlns:m="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1510040845</m:ClientId>
  <m:Package>Some article number</m:Package>
  <m:Result>SUCCESS</m:Result>
</m:DeactivateFeaturePackageResponse>
```

Result can either be “ERROR” or “SUCCESS” or “PENDING”.

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”, “RendererState”, “WolInteractive”, “NetworkHostName”, “TvServiceAudioIndex”, “TvServiceSubtitleIndex”.

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

gives a response

```
<m:GetSettingsResponse>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:MultiroomActive>1</m:MultiroomActive>
 <m:WolEnable>0</m:WolEnable>
 <m:WolInteractive>1</m:WolInteractive>
 <m:RemoterecEnabled>1</m:RemoterecEnabled>
 <m:RemoterecEmailConfigurationType>0</m:RemoterecEmailConfigurationType>
 <m:RemoterecScanStartOfPeriod>0</m:RemoterecScanStartOfPeriod>
 <m:RemoterecScanEndOfPeriod>21600</m:RemoterecScanEndOfPeriod>
 <m:RemoterecEmailAddress>tv-xxxxxxxxxxxxxxx@loewe-dialogue.com</m:RemoterecEmailAddress>
 <m:RemoterecEmailSecurePin>1234</m:RemoterecEmailSecurePin>
 <m:StreamingShareServer>1</m:StreamingShareServer>
 <m:RendererState>1</m:RendererState>
 <m:NetworkHostName>Loewe Connect 55 – da14</m:NetworkHostName>
 <m:TvServiceAudioIndex>2</m:TvServiceAudioIndex>
 <m:TvServiceSubtitleIndex>-1</m:TvServiceSubtitleIndex>
</m:GetSettingsResponse>
```

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

gives a response

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

MultiroomActive	Multiroom status.	0 (disabled), 1 (enabled)
WolEnable	Wake on LAN status.	0 (disabled), 1 (enabled)
WolInteractive	Wake up to interactive mode status.	0 (disabled), 1 (enabled)
RemoterecEnabled	Remote recording status.	0 (disabled), 1 (enabled)
RemoterecEmailConfigurationType	Remote recording email configuration type.	0 (POP3), 1 (IMAP)
RemoterecScanStartOfPeriod	Start of email scan period in seconds from midnight.	0 -86400
RemoterecScanEndOfPeriod	End of email scan period in seconds from midnight.	0 -86400
RemoterecEmailAddress	Remote recording email address.	
RemoterecEmailSecurePin	Remote recording email pin, parsed for identification.	
StreamingShareServer	Streaming share server status.	0 (disabled), 1 (enabled)
RendererState	Renderer status.	0 (disabled), 1 (enabled)
NetworkHostName	Network host name of the TV.	

TvServiceAudioIndex	Audio/language selection ID of the active channel. An iterator can be created to get more information about the selection using the ID.	
TvServiceSubtitleIndex	Subtitle selection ID of the active channel. An iterator can be created to get more information about the selection using the ID.	

9.20 Query (DR+ archive) volumes

A volume is a partition, which is used to group files. There are many specialized types of volumes, for example the volume of the DR+ archive of the integrated hard disk drive or a DR+ archive shared by network. This query can give all volumes known to the TV. A filter can be set to query for a certain type of volume.

```
<m:GetVolumes>
  <m:fcid>4138941</m:fcid>
  <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
  <m:QueryParameters>
    <m:Range startIndex="100" maxItems="100"/>
    <m:OrderField field="userChannelNumber" type="ascending"/>
  </m:QueryParameters>
  <m:AdditionalParameters>
    <m:Properties hasPurpose="0"/>
  </m:AdditionalParameters>
</m:GetVolumes>
```

Range		Not supported.	
OrderField		Not supported.	
Properties	hasPurpose	Bitmask of volume purpose flags. Results have all bits of the mask set.	See below.

Table of volume purpose flags

0x00000001	Volume used internally in the system.
0x00000002	Volume is integral part of a sealed box product.
0x00000004	Volume is externally attached to a product.
0x00000008	Volume has been verified to be a genuine manufacturer's product.
0x00000010	Volume refers to something connected via network.
0x00000020	Volume refers to something via logical file system. (fsal2)
0x00000100	Volume can be used as a source for system updates.
0x00000200	Volume can be used as storage memory for PVR archive storage.
0x00000400	Volume can be used as a temporary storage memory for timeshifting.

0x00001000	Volume can be used as a media source.
0x00002000	Volume can be used as storage for media files.
0x00010000	Volume is especially suited for storage of image files.
0x00020000	Volume is especially suited for storage of audio files.
0x00040000	Volume is especially suited for storage of video files.
0x00100000	Volume accepts browse queries.
0x00200000	Volume accepts search queries.
0x00400000	Volume is externally layouted.
0x00800000	Volume accepts search queries for PVR.

Example: hasPurpose="0x200" gives internally and externally connected DR+ archive volumes. hasPurpose="0x204" gives just the externally connected DR+ archive volumes. HasPurpose="0" gives all volumes known to the device.

```
<m:GetVolumesResponse>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:ResultVolumesFragment totalResults="13">
  <m:ResultVolume>
   <m:Id>FSL2://0f60601b-dc63-482a-acdd-521526134cdf</m:Id>
   <m:Name>#3058</m:Name>
   <m:Purpose>11535912</m:Purpose>
  </m:ResultVolume>
 </m:ResultVolumesFragment>
</m:GetSettingsResponse>
```

ResultVolumeFragment	totalResults	Number of all volumes known to the device.	
ResultVolume	Id	Unique Id of the volume result. Can be used as a parameter for GetDRPlusArchive.	
ResultVolume	Name	Name displayed in GUI.	
ResultVolume	Purpose	Bitmask of volume purposes in decimal.	

9.21 Parental Lock control

9.21.1 Set parental lock setting

Set some parental lock settings. Some steps of the parental lock configuration, such as setting the pin code, have to be performed in the TV menu.

```
<m:ParentalLock>
  <m:fcid>4138941</m:fcid>
  <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
  <m:LockParameters>
    <m:LockAll pin="" lockAll="" lockAllDaily="" lockAllDaily="" lockAllDailyStart="" lockAllDailyEnd="" />
    <m:LockSingle pin="" lockSingle=""/>
    <m:LockAgeRelated pin="" lockAgeRelated="" lockAge=""/>
  </m:LockParameters>
  <m:LocatorSequence>
    <m:Locator Locator="channel://1:ch7e3fefa0-39ba-4e36-8f33-0d1b3edd754e:y"/>
    <m:Locator Locator="channel://3:ch5e4aa534-f26d-467b-b09f-25d2a674380d:y"/>
  </m:LocatorSequence>
</m:ParentalLock>
```

LockAll	pin	Mandatory to be able to perform parental lock on all stations.	4 digits
LockAll	lockAll	Lock or unlock all stations.	0 (unlock), 1 (lock)
LockAll	lockAllDaily	Set daily time period for which device is locked.	0 (off), 1 (on)
LockAll	lockAllDailyStart	Start time of daily lock time period in seconds.	0 - 86340
LockAll	lockAllDailyEnd	End time of daily lock time period in seconds.	0 - 86340
LockSingle	pin	Mandatory to be able to perform parental lock on single stations.	4 digits
LockSingle	lockSingle	Lock or unlock single stations. Stations set in LocatorSequence.	0 (unlock), 1 (lock)
LockAgeRelated	pin	Mandatory to be able to perform age-related parental lock on all stations.	4 digits
LockAgeRelated	lockAgeRelated	Lock or unlock all stations age-related.	0 (unlock), 1 (lock)
LockAgeRelated	lockAge	Set age for age-related lock.	3 - 18
Locator	Locator	Locator URI of station to be locked by lockSingle attribute. Can be obtained from GetChannelList and others.	

```
<m:ParentalLockResponse>
 <m:fcid>8138436</m:fcid>
 <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
 <m:Result>OK</m:Result>
</m:ParentalLockResponse>
```

9.21.2 Get parental lock setting

Get parental lock settings. See ParentalLock for a description of the settings.

```
<m:GetParentalLock>
  <m:fcid>4138941</m:fcid>
  <m:ClientId>LRemoteClient-0-1314017969</m:ClientId>
</m:GetParentalLock>
```

```
<m:GetParentalLockResponse>
  <m:lockAll>0</m:lockAll>
  <m:lockAllDaily>0</m:lockAllDaily>
  <m:lockAllDailyStart>0</m:lockAllDailyStart>
  <m:lockAllDailyEnd>0</m:lockAllDailyEnd>
  <m:lockAgeRelated>0</m:lockAgeRelated>
  <m:lockAge>18</m:lockAge>
</m:GetParentalLockResponse>
```

9.22 Iterators

Iterators are used to traverse the table containers used for station lists, EPG, recordings, recording timers, dashboard entries, among others. In most cases the iterator has to be created with a certain selection of properties, to get exactly the results fitting these properties. We already have some specialized SOAP functions, which use iterators internally to gather their results, such as GetChannelList, GetListOfChannelLists, GetMediaItem, GetDRPlusArchive, GetTimerList, among others. The following SOAP functions for iterators expose an extensive, internal API and a description of all the possible selection values etc. would go beyond the scope of this document. Please ask if you want to use this.

9.22.1 Create iterator

Create an iterator and get specific results from the table containers. Table name, iterator type, selections, fields and orders are mandatory. NotifyUpdate, QueryParameters and SeekToContentParameters are optional. NotifyUpdate is used to signal when some value held by the iterator has changed for the first time and the results received previously have to be considered deprecated. Upnp Notifications are used for this. QueryParameters can be used to limit the number of results, by setting a start index and the maximum number of results. SeekToContentParameters can also be used to limit the number of results, by seeking to the iterator position matching some specific selection.

The following example creates an iterator to get exactly the first EPG entry (QueryParameters Range maxItems="1") associated to the service with DVB triplet 10301 (service ID), 1019 (transport stream ID), 1 (original network ID) with a start time greater than 1509973200 (UTC represented as Unix time). In the reponse it is noted that this entry is at index 41 of a total of 864 entries.

```
<m:CreateIterator>
 <fcid>?</fcid>
 <ClientId>LRemoteClient-0-1509972872</ClientId>
 <TableName>de.loewe.sl2.epg</TableName>
 <Type>0</Type>
 <Selections>
   <Selection>
    <Field>2002</Field>
            <Condition>1</Condition>
    <Value>1</Value>
  </Selection>
  <Selection>
    <Field>2000</Field>
            <Condition>1</Condition>
    <Value>10301</Value>
  </Selection>
     <Selection>
    <Field>2001</Field>
            <Condition>1</Condition>
    <Value>1019</Value>
  </Selection>
 </Selections>
 <Fields>
   <Field>2000</Field>
   <Field>2001</Field>
   <Field>2002</Field>
   <Field>2004</Field>
   <Field>2007</Field>
 </Fields>
 <Orders>
   <Order>
     <Field>2000</Field>
     <Direction>1</Direction>
   </Order>
 </Orders>
 <NotifyUpdate>NO</NotifyUpdate>
 <QueryParameters>
   <Range startIndex="41" maxItems="1"/>
   <MediaItemInformation>?</MediaItemInformation>
   <MediaItemClass>?</MediaItemClass>
 </QueryParameters>
 <SeekToContentParameters>
   <Selections>
     <Selection>
       <Field>2007</Field>
            <Condition>5</Condition>
       <Value>1509973200</Value>
     </Selection>
   </Selections>
   <Which>0</Which>
   <Offset>0</Offset>
   <WindowSize>0</WindowSize>
 </SeekToContentParameters>
</m:CreateIterator>
```

```
<m:CreateIteratorResponse xmlns:ltv="urn:loewe.de:RemoteTV:Tablet">
<m:fcid>?</m:fcid>
<m:ClientId>LRemoteClient-0-1509965273</m:ClientId>
<m:Result>0</m:Result>
<m:Id>1509971157234</m:Id>
<m:Rows totalResults="864" returnedResults="1" startIndex="41">
  <m:Row>
    <m:V>10301</m:V>
    <m:V>1019</m:V>
    <m:V>1</m:V>
    <m:V>Rote Rosen (2539)</m:V>
    <m:V>1509973800</m:V>
  </m:Row>
</m:Rows>
</m:CreateIteratorResponse>
```

9.22.2 Close iterator

When NotifyUpdate is set to “YES”, the iterator is kept in memory indefinitely, to be able to receive and signal the update. If a client does not need to receive this update it can and should close the iterator manually using the ID, to free up resources. Resources are freed up automatically when the iterator receives an update or after it has been created with NotifyUpdate “NO”.

```
<m:CloseIterator>
  <fcid>?</fcid>
  <ClientId>LRemoteClient-0-1502803780</ClientId>
  <Ids>
    <Id>1509971157234</Id>
  </Ids>
</m:CloseIterator>
```

```
<m:CloseIteratorResponse xmlns:ltv="urn:loewe.de:RemoteTV:Tablet">
  <m:fcid>?</m:fcid>
  <m:ClientId>LRemoteClient-0-1502803780</m:ClientId>
  <m:Results>
    <m:Result>
      <m:Id>1509971157234</m:Id>
      <m:IsClosed>Yes</m:IsClosed>
    </m:Result>
  </m:Results>
</m:CloseIteratorResponse>
```

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

Impressum

Loewe Technologies GmbH
Industriestraße 11
D-96317 Kronach
www.loewe.tv

Redaktionsdatum 03/18 TH
© Loewe Technologies GmbH, Kronach

Die Informationen in diesem Dokument können sich ohne vorherige Ankündigung ändern und stellen keinerlei Verpflichtung seitens der Loewe Technologies GmbH dar.

Alle Rechte, auch die der Übersetzung, technische Änderungen sowie Irrtümer vorbehalten.