Table of Contents

TABLE OF CONTENTS  3

PART 1: USER MANUAL  6

1. OVERVIEW  7
  1.1. PURPOSE OF THIS DOCUMENT ................................................................. 7
      1.1.1. About the Khronos Group ............................................................. 7
  1.2. INTENDED AUDIENCE ........................................................................ 7
  1.3. THE OPENMAX AL DIGITAL TV EXTENSION ............................................. 8
  1.4. ACKNOWLEDGEMENTS ...................................................................... 9

2. OPENMAX AL DIGITAL TV EXTENSION FEATURES  10

3. DESIGN OVERVIEW  11

4. FUNCTIONAL OVERVIEW  12
  4.1. OBJECT OVERVIEW ........................................................................... 12
      4.1.1. Digital TV Source Object ............................................................... 12
      4.1.2. Program Guide Manager Object .................................................. 13
      4.1.3. Service object ............................................................................ 13
  4.2. MINIMUM REQUIREMENTS ..................................................................... 14
  4.3. OPENMAX AL DIGITAL TV EXTENSION USECASES .............................. 15
      4.3.1. Live Broadcasted Playback ............................................................ 15
      4.3.2. Live Broadcasted Playback with Time Shift ..................................... 16
      4.3.3. Live Broadcasted Playback with Time Shift (alternative) ................. 17
      4.3.4. Live Broadcasted Playback with Time Shift and Recording .............. 18
      4.3.5. Live Broadcasted Playback — Switching Service ............................ 19
      4.3.6. Record Broadcasted Content ......................................................... 20
      4.3.7. Simultaneous Record and Playback of Broadcasted Content ............ 21
      4.3.8. Retrieval and Query of the Electronic Program Guide (EPG) .......... 22
      4.3.9. Live Playback Initiated by the EPG ............................................... 23
      4.3.10. Data Delivery ........................................................................... 24
      4.3.11. Purchasing Initiation ................................................................ 25
  4.4. MEMORY MANAGEMENT .................................................................... 26

5. ENGINE EXTENSIONS  27

PART 2: API REFERENCE  28

6. OBJECT DEFINITIONS  29
6.1. DTVSource Object .......................................................................................... 29
6.2. Player Object ............................................................................................... 30
6.3. Program Guide Manager Object .................................................................... 31
6.4. Recorder Object .......................................................................................... 32
6.5. Service Object ............................................................................................. 33

7. Interface Definitions 34

7.1. XADTVContentProtectionITF ....................................................................... 35
7.2. XADTVPlayerTimedObjectsITF ................................................................. 37
7.3. XADTVPlayerTimeShiftControlITF ............................................................. 42
7.4. XADTVProgramGuidePurchaseITF ............................................................. 48
7.5. XADTVProgramGuideQueryITF ................................................................. 50
7.6. XADTVProgramGuideUpdateITF ................................................................. 71
7.7. XADTVRecorderModeITF ........................................................................... 80
7.8. XADTVServiceDataDeliveryITF .................................................................. 83
7.9. XADTVServiceInputSelectorITF ................................................................ 91
7.10. XADTVSourceBroadcastITF ..................................................................... 94
7.11. XADTVSourceLocalITF ........................................................................ 101
7.12. XADTVSourceMulticastITF .................................................................. 104
7.13. XADTVSourceUnicastITF ..................................................................... 109
7.14. XADTVSourceUtilitiesITF ..................................................................... 114

8. Macros and TypeDefs 123

8.1. Structures ................................................................................................ 123
8.1.1. XADataLocator_DTVService ............................................................... 123
8.1.2. XADTVContentProtectionInfo .......................................................... 123
8.1.3. XADTVInternetConnectionInfo ......................................................... 123
8.1.4. XADTVLocalAssociateInfo ............................................................... 124
8.1.5. XADTVMulticastGroupInfo ............................................................... 124
8.1.6. XADTVServiceConnectionInfo .......................................................... 124
8.1.7. XADTVServiceDataDeliveryFileDescriptor ...................................... 125
8.1.8. XADTVServiceTimeShiftInfo ............................................................. 125
8.1.9. XADTVSourceBearerInfo ................................................................. 126
8.1.10. XADTVSourceScanInfo .................................................................. 126
8.1.11. XADTVUnicastConnectInfo ............................................................. 127
8.2. Macros .................................................................................................... 129
8.2.1. XA_DTV_EPG_ATTR_INT ............................................................... 129
8.2.2. XA_DTV_EPG_ATTR_INT_PRICE .................................................. 129
8.2.3. XA_DTV_EPG_ATTR_STR ............................................................... 129
8.2.4. XA_DTV_EPG_ATTR_TIME ............................................................. 130
8.2.5. XA_DTV_EPG_OPR_COMP ............................................................. 130
8.2.6. XA_DTV_EPG_PERMISSION .......................................................... 130
8.2.7. XA_DTV_BEARER_FAMILY ............................................................. 131
8.2.8. XA_DTV_BEARER_STATE ............................................................. 131
8.2.9. XA_DTV_BEARER_TECHNOLOGY .................................................... 131
8.2.10. XA_DTV_BEARER_TYPE ............................................................. 132
8.2.11. XA_DTV_CONTENT_PROTECTION ................................................... 132
8.2.12. XA_DTV_EPG_CONTENT_ID_TYPE ............................................... 133
8.2.13. XA_DTV_MULTICAST_CONNECTION_STATE ........................................ 133
8.2.14. XA_DTV_MULTICAST_TECHNOLOGY ........................................... 133
8.2.15. XA_DTV_PLAYER_TIMED_OBJECTS ............................................. 134
8.2.16. XA_DTV_RECORDER_MODE ....................................................... 134
8.2.17. XA_DTV_SERVICE_FILE_DL_PROGRESS .......................................... 134
8.2.18. XA_DTV_SERVICE_FILE_DESCRIPTION ........................................... 135
8.2.19. XA_DTV_SERVICE_TYPE ............................................................ 135
8.2.20. XA_DTV_SOURCE_BEARER_CHANGE ............................................ 136
8.2.21. XA_DTV_UNICAST_CONNECTION_STATE ....................................... 136
8.2.22. XA_DTV_UNICAST_TECHNOLOGY ............................................... 136

PART 3: APPENDICES 138

APPENDIX A: REFERENCES 139

APPENDIX B: OPENMAX AL PROFILE-OBJECT MAPPING 140

APPENDIX C: OBJECT-INTERFACE MAPPING 141
1. Overview

1.1. Purpose of this Document

This document details the API for an extension to OpenMAX Application Layer (AL)
1.1. Developed as an open standard by the Khronos Group, OpenMAX AL Digital TV
Extension is an application-level Digital TV API. It provides a device-independent,
cross-platform interface for applications to access a device’s TV capabilities.

1.1.1. About the Khronos Group

The Khronos Group is a member-funded industry consortium focused on the
creation of open standard, royalty-free APIs to enable the authoring and accelerated
playback of dynamic media on a wide variety of platforms and devices. All Khronos
members can contribute to the development of Khronos API specifications, are
empowered to vote at various stages before public deployment, and may accelerate
the delivery of their multimedia platforms and applications through early access to
specification drafts and conformance tests. The Khronos Group is responsible for
open APIs such as OpenGL ES, OpenKODE, OpenSL ES and OpenVG.

1.2. Intended Audience

This specification is meant for application-developers and implementers. The
document is split into a user manual section and an API reference section.
Application-developers can use this document as a user guide to learn about how to
use OpenMAX AL Digital TV extension and they can refer to the API reference when
developing their applications. Implementers of the API can use this specification to
determine what constitutes an implementation conforming to the OpenMAX AL
Digital TV extension standard.
1.3. The OpenMAX AL Digital TV Extension

The OpenMAX AL Digital TV Extension provides an API to control reception, recording and playback of digital TV content. This new extension introduces three new objects:

- **DTVSource object** which abstracts the tuner hardware. DTVSource includes bearer selection, connections to streaming servers and scanning functionality.
- **Program Guide Manager object** which provides access to Electronic Program Guides (EPGs).
- **Service object** which represents the content received. Content received could be a TV channel. Service objects act as a data source for both the Player and the Recorder.

In addition to those three new objects a number of extension interfaces have been added to the OpenMAX AL player and recorder objects.

The Digital TV Extension requires conforming to the Media Player profile of the OpenMAX AL specification. If Digital TV Recording capabilities are to be implemented the Media Player/Recorder profile is mandated on the OpenMAX AL implementation.

![Figure 1 - Overview](image-url)
1.4. Acknowledgements

The OpenMAX AL Mobile TV extension specification is the result of the contributions of many people. The following is a partial list of contributors in order of company name then contributor's surname, including the respective companies represented at the time of their contribution:

Robert Alm, Ericsson (Editor)
Göran Roth, Ericsson
Harald Gustafsson, Ericsson
Håkan Gårdrup, Ericsson
Erik Noreke, Ericsson
Joakim Roubert, Ericsson
Yeshwant Muthusamy, Nokia
Matti Paavola, Nokia
Isaac Richards, NVIDIA
Jim Van Welzen, NVIDIA
Tom Longo, Qualcomm
Steven Winston, SRS Labs
2. OpenMAX AL Digital TV extension features

The Digital TV extension provides technology neutral access to broadcast, unicast and multicast delivery methods for Digital TV. The extension is modularized by using objects to represent each part of a full Digital TV implementation.

The extension's modules and their high level intended capabilities are:

- Service Object – Acts as data source for the player. It uses the Digital TV Source object to retrieve data.
- Digital TV Source – Communicates with the reception hardware and streaming servers. This object can handle broadcasts, multicasts and unicasts.
- Program Guide manager – Handles queries on the program guide, and provides service information to the application.

Of these objects only the Service and the Digital TV Source objects are mandated, as those objects act as data source for the player. However, it is highly encouraged to implement the program guide manager object to get a feature rich Digital TV implementation. If program guides are to be used, the program guide manager needs to be implemented.
3. Design Overview

The Digital TV Extension uses an object oriented approach, extending the OpenMAX AL design. For specifics about how objects are handled and created, refer to the OpenMAX AL specification's section on object manipulation.
4. Functional Overview

4.1. Object Overview

The OpenMAX AL Digital TV Extension provides the following objects:

- Digital TV Source object
- Program Guide Manager objects
- Service objects
- Extended Standard OpenMAX AL Player object
- Extended Standard OpenMAX AL Recorder object
- Extended Standard OpenMAX AL Engine object

The objects and their interfaces are illustrated in the figure below.

![Figure 2 - All objects and interfaces](image)

The following sections provide information about each of these objects. Please refer to section 7 for more information about the individual interfaces.

4.1.1. Digital TV Source Object

The Digital TV Source object is the data provider for the source objects. All reception of the digital TV services is handled in this object. The object provides interfaces that are able to enumerate all available reception technologies in the device and perform service scanning on them.

In most uses only one instance of this object is needed, however some specially crafted applications might want to have more than one instance of this object. The implementation does not disallow creating multiple instances of the object. What needs to be taken in consideration is the fact that some hardware tuners can only be used on one service at a time. The limitation on how many services a receiver
can be bound to is hardware specific.
First, received data is paired with Bearer IDs. Then, a service can use the bearer’s ID to retrieve the received data.

4.1.2. Program Guide Manager Object

The Program Guide Manager object provides interfaces to query data from the Program guide associated with the tuners and services available. The retrieved data from queries on this object can be used to instantiate services.

4.1.3. Service object

The Service object represents a single Digital TV Service; however, multiple service objects may be instantiated to represent multiple services simultaneously.

The service object acts as the data source for the player or recorder objects via the OpenMAX AL XADataSource structure. A new data locator, XADataLocator_DTVService (see Section 8.1.1), is used for this purpose. For further information on how to use data locators and data sources refer to the OpenMAX AL specification. One service may be recorded while watching another.

The service object can be associated with a Digital TV Source object using two different connection methods. Each of the two methods has its own advantage.

- Connected via information retrieved from the program guide manager. Advantage: Easy to connect to a queried program guide service.
- Connected directly to a pre-tuned receiver bearer on the Digital TV Source object. Advantage: Simple connection for broadcast not using a program guide.
4.2. **Minimum Requirements**

Apart from the mandatory interfaces (See Appendix C:) that need to be implemented, there exist a couple of minimum requirements on the implementation.

1. At least one reception bearer technology interface needs to be implemented.
2. The implementation needs to be able to save at least 60 seconds of transmission data to be used for the time shift functionality.
3. The implementation needs to be able to save at least 60 seconds of transmission data using the recorder.
4.3. **OpenMAX AL Digital TV Extension Usecases**

4.3.1. **Live Broadcasted Playback**

This usecase illustrates the interfaces and objects involved in setting up and using a generic live broadcasted reception technology.

The usecase illustrates a broadcast-oriented reception, but the usecase could also be applied to any other reception technology by just changing the used reception interface on the DTVSource object.

This usecase extends the OpenMAX AL standard usecase for Audio and Video Playback.

1. First create the DTV Source and Service objects. Then create the playback object as described in the Audio and Video Playback usecase from the OpenMAX AL specification.

2. Using the `XADTVSourceUtilitiesItf` from the DTV Source object, check that the reception technology you want to use is available, and then save the `bearerID`.

3. Scan the broadcast technology using the `XADTVSourceBroadcastItf` to find a suitable service (TV channel) to use.

4. Tune the bearer to the service (TV channel) that was chosen in step 3 above by using the `XADTVSourceBroadcastItf`.

5. On the service object, use the `XADTVServiceInputSelectorItf` to connect the tuned bearer using the `bearerID` as a reference.

6. Connect the service object as a data source for the player - either when creating the player or by using the `XADynamicSourceSinkChangeItf`.

7. Start the rendering by issuing a play on the Player object using the `XAPlayItf`.

---

**Figure 3 - Live Broadcasted Playback**

1. First create the DTV Source and Service objects. Then create the playback object as described in the Audio and Video Playback usecase from the OpenMAX AL specification.

2. Using the `XADTVSourceUtilitiesItf` from the DTV Source object, check that the reception technology you want to use is available, and then save the `bearerID`.

3. Scan the broadcast technology using the `XADTVSourceBroadcastItf` to find a suitable service (TV channel) to use.

4. Tune the bearer to the service (TV channel) that was chosen in step 3 above by using the `XADTVSourceBroadcastItf`.

5. On the service object, use the `XADTVServiceInputSelectorItf` to connect the tuned bearer using the `bearerID` as a reference.

6. Connect the service object as a data source for the player - either when creating the player or by using the `XADynamicSourceSinkChangeItf`.

7. Start the rendering by issuing a play on the Player object using the `XAPlayItf`.
4.3.2. **Live Broadcasted Playback with Time Shift**

This usecase illustrates the interfaces and objects involved in setting up a time shifted live playback session. Time shift is when a live transmission feed is saved locally allowing you to pause during playback.

The procedure is the same as the Live Broadcasted Playback, but the **XADTVPlayerTimeShiftControlItf** is also used.

---

**Figure 4 - Live Broadcasted Playback with Time Shift**

Time shifting is performed by initializing the time shift functionality using the **XADTVPlayerTimeShiftControlItf** and then use the normal player controls. Use Pause, Play and Stopped states on the **XAPlayItf** and utilize the seeking functionality provided by the **XASeekItf**.
4.3.3. Live Broadcasted Playback with Time Shift (alternative)

This usecase illustrates an alternative way of achieving time shift functionality, if the underlying system allows reading a file in the file system while the same file is being written by the recorder.

For the Digital TV extension usecase parts the setup is basically analogous with the Record Broadcasted Content usecase in section 4.3.6 with the addition of a normal OpenMAX AL Media Player object reading from a file source that refers to the recorded file.

Note that this usecase is only for illustrative purposes and mimics the way many other current DTV implementations do time shifting. It is up to the application implementing the end user solution to decide how time shift should be implemented. The recommendation is, however, to follow the usecase illustrated in section 4.3.2, since the XADTVPlayerTimeShiftControlItf provides functionality (e.g. time shift callback, enable/disable) that is unavailable in this use case.

See below for an illustration of this alternative concept.

Figure 5 - Live Broadcasted Playback with Time Shift (alternative)

The Player illustrated in this scenario is a standard OpenMAX AL player object.
4.3.4. Live Broadcasted Playback with Time Shift and Recording

This usecase illustrates simultaneous recording of playback and time shifted content.

The time shift functionality resides on the player object so that the time shift does not interfere with the recording. The Service outputs the same information to both the Player and the Recorder.

**Figure 6 - Live Broadcasted Playback with Time Shift and Recording**

1. Set up Live Broadcasted Playback with Time Shift by following the usecase illustrated in section 4.3.2.
2. Create a Recorder object and associate the same service with that Recorder.
3. Use the `XADTVRecorderModeItf` and the `XARecordItf` to record the service.
4.3.5. Live Broadcasted Playback — Switching Service

This usecase illustrates the interfaces and objects involved when switching service.

---

**Figure 7 - Live Broadcasted Playback - Switching Service**

1. Set up Live Broadcasted Playback by following the Live Broadcasted Playback scenario in section 4.3.1.
2. Create a new “Service 2” object that will represent the new service.
3. On the Player object issue a Stop using the XAPlayItf.
4. Destroy the “Service 1” object.
5. Using the DTV Source object tune the bearer to the new service.
6. Connect the “Service 2” object with the DTV Source using the newly tuned bearerID in the XADTVServiceInputSelectorItf.
7. Connect the “Service 2” object as DataSource to the Player using the XADynamicSourceSinkChangeItf.
8. On the Player object issue a Play using the XAPlayItf.
4.3.6. Record Broadcasted Content

This usecase illustrates the interfaces and objects involved in setting up a recording session.

This usecase is based on the Live Broadcasted Playback usecase above.

1. Follow the Live Broadcasted Playback scenario in section 4.3.1, creating a Recorder object instead of a Player object.
2. Set the recorder mode by using the XADTVRecorderModeItf interface.
3. Record the broadcast by using the XARecordItf interface.
4.3.7. Simultaneous Record and Playback of Broadcasted Content

This usecase illustrates which objects and interfaces that are needed to set up simultaneous recording and playback of the same broadcasted content.

The Digital TV Extension is not limited to recording the same service that is currently rendered. To record one service while watching another, simply create more than one service object and connect one of them to the recorder and the other one to the player.

**Figure 9 - Simultaneous Record and Playback of Broadcasted Content**

1. Follow the Live Broadcasted Playback scenario in section 4.3.1, creating a Recorder object and a Player object.
2. Connect the Player and Recorder objects to the Service using the XADynamicSourceSinkChangeItf or when creating the objects.
3. Set the recorder mode by using the XADTVRecorderModeItf interface.
4. Record the broadcast by using the XARecordItf interface.
5. Start the player rendering by using the XAPlayItf.
4.3.8. Retrieval and Query of the Electronic Program Guide (EPG)

This usecase illustrates the interfaces and objects involved in receiving and querying of the Electronic Program Guide (EPG).

1. Select and scan a broadcast bearer to find either a single service or a combined multiplex of multiple services.
2. Tune the bearer to the service or services found.
3. Use the XADTVProgramGuideUpdateItf to connect the tuned bearer using the associated bearerID. Also supply a callback function to receive notifications of updates to the EPG. It is possible to associate multiple bearers using the XADTVProgramGuideUpdateItf to a single program guide manager. Please refer to the XADTVProgramGuideUpdateItf in section 7.6.
4. The EPG downloads continuously. This might take some time depending on the technology used and amount of data contained in the EPG.
5. The Digital TV extension does not determine when to allow queries in the EPG as downloading an EPG is normally a continuous process with no end. It is up to the application to determine when to allow queries.
6. Query the EPG by using the XADTVProgramGuideQueryItf to locate service or content of interest.
7. Alternative (if a previously downloaded EPG has been saved to the file system):
8. Use the XADTVProgramGuideUpdateItf to set a previously downloaded EPG as the current EPG.
9. Query the EPG by using the XADTVProgramGuideQueryItf to locate service or content of interest.

Figure 10 - Retrieval and Query of the EPG
4.3.9. **Live Playback Initiated by the EPG**

This usecase illustrates the interfaces and objects involved in setting up a live playback session using the program guide. The usecase also illustrates the events that are triggered when trying to render protected content without proper rights.

1. Retrieve and query an EPG by following the “Retrieval and Query of the EPG” usecase in section 4.3.8.

2. Gather the needed connection information by using the XADTVProgramGuideQueryItf.

3. Connect the Service object by using the XADTVServiceInputSelectorItf. The Service object will automatically utilize the DTV Source to set up the correct hardware resources needed.

4. When the connection is established and the bearer is tuned the service object is ready to be used.

5. For the application to be able to get notifications of content protected streams or multiplexes the XADTVContentProtectionItf needs to be used and a callback method must be registered. The XADTVContentProtectionItf is available on the Player, Recorder and Service objects. Any of those objects may handle the content protection at any time. The application must register for content protection callbacks if those callbacks are of interest. The callback event generated due to missing valid rights contains a purchase URI linking to where to acquire rights.

6. Connect the Player object using the XADynamicSourceSinkChangeItf.

7. Set the player in XA_PLAYSTATE_PLAYING state by using the XAPlayItf.
4.3.10. Data Delivery

This usecase illustrates the interfaces and objects involved in setting up a data delivery session.

Data delivery is the technology used by many operators to distribute extra binary data to broadcast receivers and not all reception technologies support data delivery.

Figure 12 - Data Delivery

1. Scan a reception technology to find a suitable service that contains data delivery. Tune the bearer to that service and create a service object.
2. Connect the service object to the tuned bearer using the XADTVServiceInputSelectorItf interface.
3. Register the content protection callback function.
4. Set up and use the Data Delivery mechanism by using the XADTVServiceDataDeliveryItf interface.
5. If the data received is content protected and no valid rights exist a callback event is generated containing a purchase URI that specifies where to acquire rights.
4.3.11. Purchasing Initiation

This usecase illustrates the interfaces and objects involved when initiating purchase of rights. The user receives a URI to a web service where to purchase the needed rights.

1. Follow the "Retrieval and Query of the Electronic Program Guide (EPG)" usecase to retrieve a program guide.
2. Query the EPG to retrieve a service ID or content ID for content that is protected.
3. Use the XADTVProgramGuidePurchaseItf to retrieve the purchase URI. The URI supplied to the application can be used to access a webpage where the required rights needed for the stream can be purchased. The application initiates the purchasing of the required rights in cooperation with the user. The Digital TV Extension is not involved during the purchasing transaction. The purchase URI is also returned when either the Service object or the Player object tries to render protected content without valid rights.
4.4. Memory management

4.4.1. General

All memory to be used by the application is allocated by the application. If there are exceptions to this, it is stated in the individual methods or interfaces.

4.4.2. Callbacks

Memory that exposes information to the application in callbacks is allocated by the implementation. If the application wants to save the information from the callback it must copy the data to its own buffers before the callback routine ends. The implementation might deallocate the buffers used in the callback when the callback routine ends.
5. Engine extensions

This section defines the additional engine extensions needed to be able to implement the Digital TV extension.

**XAStreamTypeHandlingCapabilitiesItf**

This interface is defined as an official Khronos extension interface on the engine. Please refer to that extension for more information. This extension interface is a prerequisite for the Digital TV extension.

This interface adds methods to be used to query the engine of the supported stream types.

A typical scenario for using this interface is to be able to filter out multiplexes that are unsupported by the implementation before they are shown as available to the user. This saves the application some trial-and-error attempts using the unsupported multiplexes.

In the ScanInfo structure returned during a scan there exists a MIME-type field identifying the multiplex. Cross-check this MIME-type with the capabilities returned using the **XAStreamTypeHandlingCapabilitiesItf**.
PART 2: API REFERENCE
6. Object Definitions

The new DTV-specific (DTVSource, Program Guide Manager and Service) objects below are created using the CreateExtensionObject engine method in the EngineItf, defined in the OpenMAX AL specification.

6.1. DTVSource Object

Description
This object holds functionality related to the Digital TV Sources available and poses an abstract interface towards bearer technologies, keeping them agnostic to future changes and additions.
The DTVSource object exposes multiple interfaces, each representing a reception technology or utility methods.
Besides the reception technology interfaces there exists a generic utility interface to be able to perform power management and retrieve device capabilities.
Each bearer, regardless of technology, is identified with a unique ID.
This object is mandated.

Mandated Interfaces
- XAObjectItf [See the OpenMAX AL Specification]
- XADynamicInterfaceManagementItf [See the OpenMAX AL Specification]
- XADTVSourceUtilitiesItf [See section 7.14]
And at least one of the following:
- XADTVSourceMulticastItf [See section 7.12]
- XADTVSourceUnicastItf [See section 7.13]
- XADTVSourceLocalItf [See section 0]
- XADTVSourceBroadcastItf [See section 7.10]

Applicable Optional Interfaces
- XAConfigExtensionsItf [See the OpenMAX AL Specification]
6.2. Player Object

Description

The Player object is a standard OpenMAX AL media player object on which some extension interfaces have been added.

Three extension interfaces have been added:

Timed objects (optional functionality): Handles objects to be rendered on top or next to the video. An example of this would be timed text or subtitling.

Time shift (mandated functionality): Handles the time shift functionality for Digital TV and thus enabling pause in live streams by buffering incoming data.

Content protection (mandated functionality): returns callbacks when playback cannot continue due to missing or corrupt keys.

Due to the nature of live playback the standard XAPlayItf and XASeekItf behavior varies depending on if time shift is active or not. If time shift is not initialized the only usable playstates on the XAPlayItf are XA_PLAYSTATE_PLAYING and XA_PLAYSTATE_STOPPED states. All other states return XA_RESULT_PERMISSION_DENIED.

The XASeekItf is different and will return XA_RESULT_FEATURE_UNSUPPORTED if time shift is not active, but the interface will work as normal if time shift is active. Seeking while time shift is not active is not possible.

Mandated Interfaces

All the mandated interfaces on the standard player object [See the OpenMAX AL Specification]

XADynamicSourceSinkChangeItf [See the OpenMAX AL Specification]

XADTVPlayerTimeShiftControlItf [See section 7.3]

XADTVContentProtectionItf [See section 7.1]

Applicable Optional Interfaces

XACfgExtItf [See the OpenMAX AL Specification]

XADTVPlayerTimedObjectsItf [See section 7.2]
6.3. Program Guide Manager Object

Description
This object represents the Electronic Program Guide (EPG) for the Digital TV extension.
The object contains the interfaces to set up and retrieve information, query the program guide associated with a DTV source.
This object is not mandated.

Mandated Interfaces

- XAOBJECTItf [See the OpenMAX AL Specification]
- XADYNAMICINTERFACEMANAGEMENTItf [See the OpenMAX AL Specification]
- XADTVPROMGUIDEUPDATEItf [See section 7.6]
- XADTVPROMGUIDEPURCHASEItf [See section 7.4]
- XADTVPROMGUIDEQUERYItf [See section 7.5]

Applicable Optional Interfaces

- XACONFIGEXTENSIONSItf [See the OpenMAX AL Specification]
6.4. Recorder Object

Description
The Recorder object is a standard OpenMAX AL media recorder object which has been extended with an additional extension interface that adds functionality related to recording digital TV transmissions.

The stream information interface and content protection interface as defined on the player object are also mandated for the recorder object when recording digital TV transmissions.

This object is not mandated.

Mandated Interfaces
All the mandated interfaces on the standard recorder object [See the OpenMAX AL Specification]
XADTVRecorderModeItf [See section 7.7]
XADTVContentProtectionItf [See section 7.1]

Applicable Optional Interfaces
XAConfigExtensionsItf [See the OpenMAX AL Specification]
6.5. Service Object

Description
The service object is an abstraction of a TV channel (called service in this
document). The Service object acts as a media player source. The application can
use this object to tell the DTVSource what bearer technology to use.

The service receives the content from the DTVSource, hides the underlying
decryption implementation, and sends the content to a player and/or a recorder.
The service also allows the application to access extended data, such as pod-casts
and the generic file delivery data carousel.

The criteria for adding an interface to the service object is that it affects the shared
content for all consumers. All other interfaces reside on the player or recorder.

There is one service object per content received.

This is a mandatory object.

Mandated Interfaces
- XAObjectItf [See the OpenMAX AL Specification]
- XADynamicInterfaceManagementItf [See the OpenMAX AL Specification]
- XADTVServiceInputSelectorItf [See section 7.9]
- XADTVContentProtectionItf [See section 7.1]

Applicable Optional Interfaces
- XAConfigExtensionsItf [See the OpenMAX AL Specification]
- XADTVServiceDataDeliveryItf [See section 7.8]
7. Interface Definitions

This section documents all the interfaces and methods in the API.

Almost all methods generate result codes, whether synchronously or asynchronously. Such methods must return either one of the explicit result codes listed in the method’s documentation or one of the following result codes:

XA_RESULT_RESOURCE_ERROR
XA_RESULT_RESOURCE_LOST
XA_RESULT_INTERNAL_ERROR
XA_RESULT_UNKNOWN_ERROR
XA_RESULT_OPERATION_ABORTED

See the OpenMAX AL specification for a full definition of these result codes.
7.1. XADTVContentProtectionItf

Description

XADTVContentProtectionItf is used to monitor and control the content protection for the service. No access to the content protection algorithms or decrypted keys is provided.

This interface exists on the service, the player and the recorder object. The interface is used on the various objects depending on what type of content protection is used.

The justification to let this interface be available on the service in addition to the player and recorder objects is to allow flexibility on where to handle content protection. Please refer to the Data Delivery usecase in section 4.3.6. Otherwise, the most common case when this interface is used is on the player or recorder objects. Depending on the service provider, it is possible that only some of the streams for a service will be encrypted, while others are unencrypted.

The callback always includes the needed purchase information from the EPG data supplied to the input selector if available. The callback returns the purchase information when applicable. There exists no explicit connection to the EPG as it is not mandated to always have the EPG instantiated.

This interface is a mandated interface on the Service (See section 6.5), Player (See section 6.2) and Recorder (See section 6.4) objects.

Prototype

XA_API extern const XAInterfaceID XA_IID_DTVCONTENTPROTECTION;

struct XADTVContentProtectionItf_;
typedef const struct XADTVContentProtectionItf_ * const * XADTVContentProtectionItf;
struct XADTVContentProtectionItf_ {
    XAresult (*RegisterContentProtectionCallback) (XADTVContentProtectionItf self, xaDTVContentProtectionCallback Callback, void * pContext);
};

Interface ID
cdae0f20-ba50-11de-8a39-0800200c9a66
**Callbacks**

### xaDTVContentProtectionCallback

```c
typedef void (XAAPIENTRY * xaDTVContentProtectionCallback) (
    XADTVContentProtectionItf Caller,
    XADTVContentProtectionInfo * pContentProtectionInfo,
    void * pContext
);
```

**Description**

Notifies the application that decryption of a service has failed.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller</td>
<td>[in] Interface for which this callback was registered.</td>
</tr>
<tr>
<td>pContentProtectionInfo</td>
<td>[in] The content protection info that might be needed by the application to successfully perform purchase of the required rights to decode the service.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in] User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Comments**

None

**See Also**

xaDTVServiceContentProtectionCallback()

---

**Methods**

### RegisterContentProtectionCallback

```c
XAResult (*RegisterContentProtectionCallback) ( 
    XADTVContentProtectionItf self, 
    xaDTVContentProtectionCallback Callback, 
    void * pContext
);
```

**Description**

Sets the callback for content protection event notifications.

No keys available to decrypt the service.

**Pre-conditions**

None

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback</td>
<td>[in] Specifies the callback method.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in] User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

None

**See Also**

xaDTVServiceContentProtectionCallback()
7.2. XADTVPlayerTimedObjectsItf

Description

XADTVPlayerTimedObjectsItf is used to select the text or object streams for the service. This interface is only functional if the corresponding stream is previously selected using the XAStreamInformationItf interface.

This interface is an optional interface on the Player (See section 6.2) object.

Prototype

XA_API extern const XAInterfaceID XA_IID_DTVPLAYERTIMEDOBJECTS;

struct XADTVPlayerTimedObjectsItf_;
typedef const struct XADTVPlayerTimedObjectsItf_ * const *
XADTVPlayerTimedObjectsItf;
struct XADTVPlayerTimedObjectsItf_ {
    XAresult (*GetCapabilities) (  
        XADTVPlayerTimedObjectsItf self,
        XUint32 * pCapabilities
    );
    XAresult (*SetEnable) (  
        XADTVPlayerTimedObjectsItf self,
        XUint32 EnableHint
    );
    XAresult (*GetEnable) (  
        XADTVPlayerTimedObjectsItf self,
        XUint32 * pEnableHint
    );
    XAresult (*RegisterTimedObjectsCallback) (  
        XADTVPlayerTimedObjectsItf self,
        xaDTVTimedObjectsCallback Callback,
        void * pContext
    );
};

Interface ID

5ab54890-c463-11de-8a39-0800200c9a66
Callbacks

### xaDTVTimedObjectsCallback

```c
typedef void (XAAPIENTRY * xaDTVTimedObjectsCallback) (
    XADTVPlayerTimedObjectsItf Caller,
    XAchar * pMimeType,
    void * pData,
    XAuint32 DataLength,
    XAmillisecond PresentationTime,
    XAuint32 PresentationDuration,
    XAuint32 PositionHintX,
    XAuint32 PositionHintY,
    void * pContext
);
```

**Description**
Notifies the application that an object needs to be presented at a certain presentation time.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller</td>
<td>Interface on which this callback was registered.</td>
</tr>
<tr>
<td>pMimeType</td>
<td>Mime type of the object.</td>
</tr>
<tr>
<td>pData</td>
<td>Pointer to the data that is to be presented.</td>
</tr>
<tr>
<td>DataLength</td>
<td>The length of the data pointed to in the pData pointer.</td>
</tr>
<tr>
<td>PresentationTime</td>
<td>Media position of suggested render time in milliseconds since start of rendering. The XAPlayItf::GetPosition method can be used to retrieve the current media position. That retrieved position can then be used, together with this parameter, to calculate when to present the timed object.</td>
</tr>
<tr>
<td>PresentationDuration</td>
<td>The duration of the object on screen.</td>
</tr>
<tr>
<td>PositionHintX</td>
<td>X position hint for where to render the object.</td>
</tr>
<tr>
<td>PositionHintY</td>
<td>Y position hint for where to render the object.</td>
</tr>
<tr>
<td>pContext</td>
<td>User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Comments**

**See Also**

XAPlayItf::GetPosition()
## Methods

### GetCapabilities

```c
XAResult (*GetCapabilities) (
    XADTVPlayerTimedObjectsItf self,
    XAuint32 * pCapabilities
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Gets information about the capability of the device. Returns whether the implementation is capable of rendering timed objects without application interaction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>None.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pCapabilities [out]</td>
<td>The capabilities of the device. Defined as a bitmask with implementation specific rendering capabilities. See the XA_DTV_PLAYER_TIMED_OBJECTS macro below for more information.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comments</th>
<th>The Digital TV implementation is not mandated to support any of the capabilities.</th>
</tr>
</thead>
<tbody>
<tr>
<td>See Also</td>
<td></td>
</tr>
</tbody>
</table>
### SetEnable

```c
XAresult (*SetEnable) (XADTVPlayerTimedObjectsItf self, XAuint32 EnableHint);
```

**Description**
Enables rendering of the timed text or object track, if any, associated with the player.

**Pre-conditions**
Existing media track associated with the player and with relevant information to render.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnableHint</td>
<td>[in]</td>
<td>Bitmask that defines what device rendering functionality to enable or disable. This parameter is allowed to be zero. If the application wants to let the Digital TV implementation render everything it is capable of, supply the XA_DTV_PLAYER_TIMED_OBJECTS_ALL value in this parameter. See the XA_DTV_PLAYER_TIMED_OBJECTS macro for more information.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
The Digital TV implementation is not mandated to support any of the capabilities.

**See Also**
- GetCapabilities
- xaDTVTimedObjectsCallback

### GetEnable

```c
XAresult (*GetEnable) (XADTVPlayerTimedObjectsItf self, XAuint32 * pEnableHint);
```

**Description**
Gets the current enabled rendering setting.

**Pre-conditions**
None

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pEnableHint</td>
<td>[in]</td>
<td>Bitmask defining the currently set rendering mode. See the XA_DTV_PLAYER_TIMED_OBJECTS macro for more information.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
### GetEnable

| See Also | SetEnable  
| GetCapabilities  
| xaDTVTimedObjectsCallback |

### RegisterTimedObjectsCallback

```c
XAResult (*RegisterTimedObjectsCallback) (  
XADTVPlayerTimedObjectsItf self,  
xaDTVTimedObjectsCallback Callback,  
void * pContext
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Registers timed objects callback.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>None</td>
</tr>
</tbody>
</table>

| Parameters | | |
| --- | --- | |
| Callback | [in] | A Callback that is going to be called each time a new object is to be rendered on top of the video. This parameter is allowed to be NULL if the application is not interested in any callbacks and the implementation is capable of rendering any of the object types specified by the EnableHint parameter. This callback is called for all types of objects. |
| pContext | [in] | User context data that is supplied when the callback method is registered. |

| Return value | The return value can be one of the following:  
| --- | --- |
| | • XA_RESULT_SUCCESS  
| | • XA_RESULT_PARAMETER_INVALID |

| Comments | The Digital TV implementation is not mandated to support any of the capabilities. |
| See Also | |
7.3. XADTVPlayerTimeShiftControlItf

Description

XADTVPlayerTimeShiftControlItf is used to control the time shift parameters for the service.

The system must be able to store at least 60 seconds of time shift data.

This interface is a mandated interface of the Player (See section 6.2) object.

Prototype

XA_API extern const XAInterfaceID XA_IID_DTVPLAYERTIMESHIFTCONTROL;

struct XADTVPlayerTimeShiftControlItf_;
typedef const struct XADTVPlayerTimeShiftControlItf_ * const *
XADTVPlayerTimeShiftControlItf;
struct XADTVPlayerTimeShiftControlItf_ {
    XAresult (*SetCacheLength) (  
        XADTVPlayerTimeShiftControlItf self,  
        XAboolean ringBuffer,  
        XAuint32 * pBufferSize  
    );
    XAresult (*RegisterTimeShiftInfoCallback) (  
        XADTVPlayerTimeShiftControlItf self,  
        xaDTVTimeShiftInfoCallback Callback,  
        XAMillisecond Period,  
        void * pContext  
    );
    XAresult (*GetCacheLength) (  
        XADTVPlayerTimeShiftControlItf self,  
        XAuint32 * pBufferSize,  
        XAboolean ringBuffer  
    );
    XAresult (*GetMaximumCacheLength) (  
        XADTVPlayerTimeShiftControlItf self,  
        XAuint32 * pMaxSize  
    );
    XAresult (*SetEnabled) (  
        XADTVPlayerTimeShiftControlItf self,  
        XAboolean Enable  
    );
    XAresult (*IsEnabled) (  
        XADTVPlayerTimeShiftControlItf self,  
        XAboolean * pEnabled  
    );
    XAresult (*SaveAvailableData) (  
        XADTVPlayerTimeShiftControlItf self,  
        XAchar * pUri  
    );
};
### Interface ID

6822a180-c463-11de-8a39-0800200c9a66

### Callbacks

#### xADTVTimeShiftInfoCallback

```c
typedef void (XAENTRY * xADTVTimeShiftInfoCallback) (
    XADTVPlayerTimeShiftControlItf Caller,
    XADTVServiceTimeShiftInfo * pTimeShiftInfo,
    void * pContext
);
```

#### Description
Notifies the current status of the time shift buffer to the application.

#### Parameters
- **Caller** [in] Interface on which this callback was registered.
- **pTimeShiftInfo** [in] A struct specifying the current time shift buffer status.
- **pContext** [in] User context data that is supplied when the callback method is registered.

#### Comments

#### See Also

### Methods

#### RegisterTimeShiftInfoCallback

```c
XAResult (*RegisterTimeShiftInfoCallback) (
    XADTVPlayerTimeShiftControlItf self,
    xADTVTimeShiftInfoCallback Callback,
    XAMillisecond Period,
    void * pContext
);
```

#### Description
Sets the callback and periodicity to receive time shift information. The callback is only induced if `SetCacheLength` and `Enable` have been called successfully.

#### Preconditions
None.

#### Parameters
- **Callback** [in] The callback
- **Period** [in] The periodicity for the callback in milliseconds.
- **pContext** [in] User context data that is supplied when the callback method is registered.
### RegisterTimeShiftInfoCallback

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| Comments | None |

### SetCacheLength

```c
XAResult (*SetCacheLength) (  
    XADTVPlayerTimeShiftControlItf self,  
    XAboolean ringBuffer,  
    XAuint32 * pBufferSize
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets the size of the time shift buffer. This is a hint from the application, and the actual size may vary depending on the implementation. The actual size of the buffer created is returned upon successful completion.</th>
</tr>
</thead>
</table>

| Pre-conditions | None. |

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ringBuffer [in] Set to TRUE creates a buffer which will overwrite information when full. If set to FALSE, the time shifting will end when the buffer is full, unless a new buffer is allocated.</td>
</tr>
<tr>
<td></td>
<td>pBufferSize [in/out] As an in parameter, the size of the buffer as requested by the application in bytes. When the call returns, this parameter holds the actual allocated size of the buffer, in bytes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
</tbody>
</table>

| Comments | None |

### GetCacheLength

```c
XAResult (*GetCacheLength) (  
    XADTVPlayerTimeShiftControlItf self,  
    XAuint32 * pBufferSize,  
    XAboolean * pRingbuffer
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Retrieves the size of the time shift buffer.</th>
</tr>
</thead>
</table>

| Pre-conditions | None. |

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pBufferSize [out] Returns the size of the time shift buffer in bytes.</td>
</tr>
</tbody>
</table>
### GetCacheLength

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pRingbuffer</td>
<td>[out]</td>
<td>Returns whether the buffer is a ring buffer or not.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

None

### GetMaximumCacheLength

```c
XAResult (*GetCacheLength) (XADTVPlayerTimeShiftControlItf self, XAuint32 * pMaxSize);
```

**Description**

Retrieves the maximum size the time shift buffer can have.

**Pre-conditions**

None.

**Parameters**

- `pMaxSize` [out] After a successful call this parameter holds the maximum size the time shift buffer can have.
<table>
<thead>
<tr>
<th>GetMaximumCacheLength</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Return value</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Comments</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SetEnabled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Pre-conditions</strong></td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Return value</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Comments</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IsEnabled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Pre-conditions</strong></td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### IsEnabled

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
None

### SaveAvailableData

```c
XAresult (*SaveAvailableData) (  
    XADTVPlayerTimeShiftControlItf self,  
    XAchar * pUri  
);
```

**Description**
Dumps the current time shift buffer to a file specified by `pFilePath`.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pUri</td>
<td>[in]</td>
<td>The file to save the data in. If the specified URI points to an existing file that file will be overwritten if not read-only.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
None
7.4. XADTVProgramGuidePurchaseItf

Description
This interface is the interface handling functionality associated with a purchase. It provides a means to get information that might be required by the application to perform purchase of rights.

This interface is a mandated interface on the Program Guide Manager (See section 6.3) object.

Prototype

```c
XA_API extern const XAInterfaceID XA_IID_DTVPROGRAMGUIDEPURCHASE;

struct XADTVProgramGuidePurchaseItf_;  
typedef const struct XADTVProgramGuidePurchaseItf * const * XADTVProgramGuidePurchaseItf;

struct XADTVProgramGuidePurchaseItf_ {  
    XAResult (*GetPurchaseURI) (  
        XADTVProgramGuidePurchaseItf self,  
        XAuint32 bearerID,  
        XAuint32 IDType,  
        XAuint32 ID,  
        Xauint32 * pURILength,  
        XAchar * pURI  
    );
};
```

Interface ID

4f664a20-ba4f-11de-8a39-0800200c9a66

Callbacks

None.
### Methods

<table>
<thead>
<tr>
<th>GetPurchaseURI</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAResult (*GetPurchaseURI) (</td>
</tr>
<tr>
<td>XADTVProgramGuidePurchaseItf self,</td>
</tr>
<tr>
<td>XAuint32 bearerID,</td>
</tr>
<tr>
<td>XAuint32 IDType,</td>
</tr>
<tr>
<td>XAuint32 ID,</td>
</tr>
<tr>
<td>Xauint32 * pURILength,</td>
</tr>
<tr>
<td>XAchar * pURI</td>
</tr>
<tr>
<td>);</td>
</tr>
</tbody>
</table>

**Description**

Gets the purchase URI associated with the supplied ID.

**Pre-conditions**

A Program guide must exist and be parsed.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>The ID of the associated bearer.</td>
</tr>
<tr>
<td>IDType</td>
<td>[in]</td>
<td>Indicates the type of ID supplied. See the XA_DTV_EPG_CONTENT_ID_TYPE macros for more information.</td>
</tr>
<tr>
<td>ID</td>
<td>[in]</td>
<td>The ID for the content, bundle or service to retrieve the URI from.</td>
</tr>
<tr>
<td>pURILength</td>
<td>[in/out]</td>
<td>The length of the NULL-terminated string containing the purchase URI. If the pURI parameter is NULL this method returns the length of the URI including NULL termination that is going to be returned. As an in-parameter this parameter indicates the length of the buffer supplied in the pURI parameter.</td>
</tr>
<tr>
<td>pURI</td>
<td>[out]</td>
<td>The NULL-terminated string containing the purchase URI.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

See Also
7.5. **XADTVProgramGuideQueryIft**

**Description**

This interface is used to query the electronic program guide for services or content of interest. The program guide is abstracted with "objects" of three different "classes": services ("TV channel"), contents ("TV program") and purchase bundles ("subscription package"). Each of these objects is represented as an ID number (service ID, content ID and bundle ID).

The main features of this interface are:

Methods to query services and contents (GetServices, GetContents)

Methods to query attributes (such as name, description, start time etc.) of services, contents and purchase bundles (e.g. GetServiceStringAttrib)

Macros for commonly used attributes (e.g. XA_DTV_EPG_ATTR_STR_NAME, XA_DTV_EPG_ATTR_STR_GENRE)

Methods to filter service and content queries based on attribute values (e.g. AndServiceFilterString)

Methods to find out to which purchase bundle a service or content belongs to (e.g. GetPurchaseBundlesOfService)

This interface is a mandated interface on the Program Guide Manager (See section 6.3) object.

**Prototype**
XA_API extern const XAInterfaceID XA_IID_DTVPROGRAMGUIDEQUERY;

struct XADTVProgramGuideQueryItf_;
typedef const struct XADTVProgramGuideQueryItf * const * XADTVServiceGuideQueryItf;

struct XADTVProgramGuideQueryItf_ {
    XAResult (*ANDServiceFilterString) (XADTVProgramGuideQueryItf self, XAuint32 stringAttribute, XAchar * pStringValue);
    XAResult (*ANDServiceFilterInt) (XADTVProgramGuideQueryItf self, XAuint32 intAttribute, XAuint32 intValue);
    XAResult (*ANDContentFilterString) (XADTVProgramGuideQueryItf self, XAuint32 stringAttribute, XAchar * pStringValue);
    XAResult (*ANDContentFilterTime) (XADTVProgramGuideQueryItf self, XAuint32 timeAttribute, XAtime timeValue, XAuint8 operation);
    XAResult (*ResetServiceFilterString) (XADTVProgramGuideQueryItf self, XAuint32 stringAttribute);
    XAResult (*ResetServiceFilterInt) (XADTVProgramGuideQueryItf self, XAuint32 intAttribute);
    XAResult (*ResetContentFilterString) (XADTVProgramGuideQueryItf self, XAuint32 stringAttribute);
    XAResult (*ResetContentFilterTime) (XADTVProgramGuideQueryItf self, XAuint32 timeAttribute);
    XAResult (*GetServices) (XADTVProgramGuideQueryItf self, XAuint32 * pNumberOfServices, XAuint32 * pServices);
    XAResult (*GetServiceStringAttrib) (XADTVProgramGuideQueryItf self, XAuint32 serviceID, XAuint32 stringAttribute, XAuint32 * pStringLength, XAchar * pStringValue);
    XAResult (*GetServiceIntAttrib) (XADTVProgramGuideQueryItf self, XAuint32 serviceID, XAuint32 intAttribute, XAuint32 * pValue);
};

XAResult (*GetContents) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 serviceID,
    XAuint32 * pNumberOfContents,
    XAuint32 * pContents
};

XAResult (*GetContentStringAttrib) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 contentID,
    XAuint32 stringAttribute,
    XAuint32 * pStringLength,
    XAchar * pStringValue
};

XAResult (*GetContentIntAttrib) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 contentID,
    XAuint32 intAttribute,
    XAuint32 * pIntValue
};

XAResult (*GetContentTimeAttrib) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 contentID,
    XAuint32 timeAttribute,
    XAtime * pTimeValue
};

XAResult (*GetPurchaseBundlesOfService) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 serviceID,
    XAuint32 * pNumberOfBundles,
    XAuint32 * pBundles
};

XAResult (*GetPurchaseBundlesOfContent) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 contentID,
    XAuint32 * pNumberOfBundles,
    XAuint32 * pBundles
};

XAResult (*GetAllPurchaseBundles) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 * pNumberOfBundles,
    XAuint32 * pBundles
};

XAResult (*GetServicesOfPurchaseBundle) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 bundleID,
    XAuint32 * pNumberOfServices,
    XAuint32 * pServices
};

XAResult (*GetContentsOfPurchaseBundle) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 bundleID,
    XAuint32 * pNumberOfContents,
    XAuint32 * pContents
};

XAResult (*GetSubscribedPurchaseBundles) {
    XDTPProgramGuideQueryIftf self,
    XAuint32 * pNumberOfBundles,
    XAuint32 * pBundles
};

XAResult (*GetBundleStringAttrib) {
Interface ID

5cbd85b0-af5c-11de-8a39-0800200c9a66

Callbacks

None.

Methods

<table>
<thead>
<tr>
<th>ANDServiceFilterString</th>
</tr>
</thead>
<tbody>
<tr>
<td>*<em>XAResult (<em>ANDServiceFilterString) (</em></em></td>
</tr>
<tr>
<td>XADTVProgramGuideQueryItf self,</td>
</tr>
<tr>
<td>XAuint32 stringAttribute,</td>
</tr>
<tr>
<td>XAchar * pStringValue</td>
</tr>
<tr>
<td><strong>);</strong></td>
</tr>
</tbody>
</table>

| Description | Adds a new string filtering criteria by combining it with the current existing criteria. Match is positive when the given `stringAttribute` is found as a substring in the given attribute's value. Used by `GetServices` method. |
| Pre-conditions | None. |
### ANDServiceFilterString

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pStringValue</td>
<td>[in] The new string to be ANDed to the existing criteria.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| Comments        | -                                                                            |

### ANDServiceFilterInt

```c
XAResult (*ANDServiceFilterInt) (XADTVProgramGuideQueryItf self, XAuint32 intAttribute, XAuint32 intValue);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Adds a new integer filtering criteria by ANDing it to the existing criteria. Match is positive when the given intAttribute equals with the given attribute's value. Used by GetServices method.</th>
</tr>
</thead>
</table>

| Pre-conditions  | None.                                                                                                                                     |

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intAttribute</td>
<td>[in] The new integer attribute to be ANDed to the existing criteria.</td>
</tr>
<tr>
<td>intValue</td>
<td>[in] The new integer to be ANDed to the existing criteria.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| Comments        | -                                                                            |
**ANDContentFilterString**

```
XAResult (*ANDContentFilterString) (XADTVProgramGuideQueryItf self,
XAUint32 stringAttribute,
XAchar * pStringValue);
```

**Description**
Adds a new string filtering criteria as ANDing it to the existing criteria. Match is positive when the given `stringAttribute` is found as a substring in the given attribute's value.

Used by `GetContents` method.

**Pre-conditions**
None.

**Parameters**
- stringAttribute `[in]` The new string attribute to be ANDed to the existing criteria.
- pStringValue `[in]` The new string to be ANDed to the existing criteria.

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

**Comments**
- 

**ANDContentFilterTime**

```
XAResult (*ANDContentFilterTime) (XADTVProgramGuideQueryItf self,
XAUint32 timeAttribute,
XAtime timeValue,
XAUint8 operation);
```

**Description**
Adds new filtering criteria as ANDing it to the existing criteria.

Used by `GetContents` method.

**Pre-conditions**
None.

**Parameters**
- timeAttribute `[in]` The new time attribute to be ANDed to the existing criteria.
- timeValue `[in]` The new time to be ANDed to the existing criteria.
- operation `[in]` Less than, equal or greater than. Please use `XA_DTV_EPG_OPR_COMP_` macros.
## ResetServiceFilterString

```c
XAresult (*ResetServiceFilterString) (
    XADTVProgramGuideQueryItf self,
    XAuint32 stringAttribute
);
```

### Description
Resets an existing string filtering criteria. Match is positive when the given `stringAttribute` is found as a substring in the given attribute's value. Used by `GetServices` method.

### Pre-conditions
None.

### Parameters
- **Self** [in] Interface self-reference.
- **stringAttribute** [in] The string attribute to reset.

### Return value
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

### Comments
- 

## ResetServiceFilterInt

```c
XAresult (*ResetServiceFilterInt) (
    XADTVProgramGuideQueryItf self,
    XAuint32 intAttribute
);
```

### Description
Resets an existing integer filtering. Match is positive when the given `intAttribute` equals with the given attribute's value. Used by `GetServices` method.

### Pre-conditions
None.

### Parameters
- **Self** [in] Interface self-reference.
- **intAttribute** [in] The integer attribute to reset.

### Return value
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

### Comments
- 

## ResetServiceFilterTime

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <code>XA_RESULT_SUCCESS</code></td>
</tr>
<tr>
<td></td>
<td>• <code>XA_RESULT_PARAMETER_INVALID</code></td>
</tr>
</tbody>
</table>

### Comments
- 

## ANDContentFilterTime

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <code>XA_RESULT_SUCCESS</code></td>
</tr>
<tr>
<td></td>
<td>• <code>XA_RESULT_PARAMETER_INVALID</code></td>
</tr>
</tbody>
</table>

### Comments
- 

## ResetServiceFilterString

```c
XAresult (*ResetServiceFilterString) (
    XADTVProgramGuideQueryItf self,
    XAuint32 stringAttribute
);
```
## ResetServiceFilterInt

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

- 

## ResetContentFilterString

```c
XAResult (*ResetContentFilterString) (XADTVProgramGuideQueryItf self, XAuint32 stringAttribute);
```

**Description**

Resets an existing string filtering. Match is positive when the given `stringAttribute` is found as a substring in the given attribute's value. Used by GetContents method.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringAttribute</td>
<td>[in] The string attribute to reset.</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

- 

## ResetContentFilterTime

```c
XAResult (*ResetContentFilterTime) (XADTVProgramGuideQueryItf self, XAuint32 timeAttribute);
```

**Description**

Resets existing filtering criteria. Used by GetContents method.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeAttribute</td>
<td>[in] The time attribute to reset.</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

-
### GetServices

Get service IDs matching the filter given by `ANDServiceFilter` method calls.

#### Preconditions

None.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pNumberOfServices</code></td>
<td>[in/out]</td>
<td>As an input, specifies the length of the <code>pServices</code> array. As an output, specifies the number of services currently available in the system and the length of the valid positions in the returned array. Returns 0 if no services are available in the system.</td>
</tr>
<tr>
<td><code>pServices</code></td>
<td>[out]</td>
<td>Array of the services currently available in the system. This parameter is populated by the call with the array of technologies, provided that the input value of <code>pNumberOfServices</code> is equal to or greater than the number of actual services. If <code>pNumberOfServices</code> is less than the number of actual input services, the error code <code>XA_RESULT_BUFFER_INSUFFICIENT</code> is returned.</td>
</tr>
</tbody>
</table>

#### Return value

The return value can be one of the following:

- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_RESULT_BUFFER_INSUFFICIENT`

#### Comments

The caller of this method is responsible for allocating the `pServices` array. To ensure that the allocated array is large enough to hold the returned data, this method can be called twice – once with `pNumberOfServices` as an output parameter (and `pServices` set to NULL), and the second time with the `pServices` array (of size `pNumberOfServices`) allocated and `pNumberOfServices` as an input parameter.
**GetServiceStringAttrib**

```c
XAResult (*GetServiceStringAttrib) (
    XADTVProgramGuideQueryItf self,
    XAuInt32 serviceID,
    XAuInt32 stringAttribute,
    XAuInt32 * pStringLength,
    XAChar * pStringValue
);
```

**Description**
Gets the value of the given string attribute on the given service. If the given stringAttribute is not supported by the implementation, XA_RESULT_FEATURE_UNSUPPORTED is returned.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>[in]</td>
<td>The service ID.</td>
</tr>
<tr>
<td>pStringLength</td>
<td>[in/out]</td>
<td>As an input, specifies the length of the pStringValue array. As an output, specifies the length of the returned string. May be zero if not sent.</td>
</tr>
<tr>
<td>pStringValue</td>
<td>[out]</td>
<td>Value of the given stringAttribute.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_FEATURE_UNSUPPORTED
- XA_RESULT_BUFFER_INSUFFICIENT

**Comments**
The caller of this method is responsible for allocating the pStringValue array.
### GetServiceIntAttrib

```c
XAResult (*GetServiceIntAttrib) (
    XDVTProgramGuideQueryItf self,
    XAuint32 serviceID,
    XAuint32 intAttribute,
    XAuint32 * pValue
);
```

**Description**

Gets the value of the given integer attribute on the given service. If the given `intAttribute` is not supported by the implementation, `XA_RESULT_FEATURE_UNSUPPORTED` is returned.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>[in]</td>
<td>The service ID.</td>
</tr>
<tr>
<td>intAttribute</td>
<td>[in]</td>
<td>May be zero if not sent.</td>
</tr>
<tr>
<td>pValue</td>
<td>[out]</td>
<td>Value of the given <code>intAttribute</code>.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_RESULT_FEATURE_UNSUPPORTED`

**Comments**

-

### GetContents

```c
XAResult (*GetContents) (
    XDVTProgramGuideQueryItf self,
    XAuint32 serviceID,
    XAuint32 * pNumberOfContents,
    XAuint32 * pContents
);
```

**Description**

Gets content IDs matching the filter given by `ANDContentFilter`.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>[in]</td>
<td>Service whose contents (&quot;programs&quot; usually) are queried. If zero, no filtering based on service is used.</td>
</tr>
</tbody>
</table>
### GetContents

<table>
<thead>
<tr>
<th>Parameter</th>
<th>In/Out</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pNumberOfContents</td>
<td>in/out</td>
<td>As an input, specifies the length of the pContents array. As an output, specifies the number of content IDs currently available in the system and the length of the valid positions in the returned array. Returns 0 if no content IDs are available in the system.</td>
</tr>
<tr>
<td>pContents</td>
<td>out</td>
<td>Array of content IDs.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_BUFFER_INSUFFICIENT

**Comments**
The caller of this method is responsible for allocating the pContents array.

### GetContentStringAttrib

```c
XAResult (*GetContentStringAttrib) (XADTVProgramGuideQueryItf self, XAUint32 contentID, XAUint32 stringAttribute, XAUint32 * pStringLength, XAchar * pStringValue);
```

**Description**
Gets the value of the given string attribute on the given content. If the given `stringAttribute` is not supported by the implementation, XA_RESULT_FEATURE_UNSUPPORTED is returned.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>In/Out</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentID</td>
<td>[in]</td>
<td>The service ID.</td>
</tr>
<tr>
<td>pStringLength</td>
<td>[in/out]</td>
<td>As an input, specifies the length of the pStringValue array. As an output, specifies the returned string length. May be zero if not sent.</td>
</tr>
<tr>
<td>pStringValue</td>
<td>[out]</td>
<td>Value of the given <code>stringAttribute</code>.</td>
</tr>
</tbody>
</table>
# GetContentStringAttrib

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_FEATURE_UNSUPPORTED</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_BUFFER_INSUFFICIENT</td>
</tr>
</tbody>
</table>

| Comments | The caller of this method is responsible for allocating the pStringValue array. |

# GetContentIntAttrib

```c
XAresult (*GetContentIntAttrib) (  
    XADTVProgramGuideQueryItf self,  
    XAuint32 contentID,  
    XAuint32 intAttribute,  
    XAuint32 * pIntValue  
);  
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Gets the value of the given integer attribute on the given content. If the given intAttribute is not supported by the implementation, XA_RESULT_FEATURE_UNSUPPORTED is returned.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Pre-conditions</th>
<th>None.</th>
</tr>
</thead>
</table>

| Parameters |       |          | |
|------------|-------|----------|%
| contentID  | [in]  | The content ID. |
| pIntValue  | [out] | Value of the given intAttribute. |

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_FEATURE_UNSUPPORTED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comments</th>
<th>-</th>
</tr>
</thead>
</table>
### GetContentTimeAttrib

```c
XAResult (*GetContentTimeAttrib) ( 
  XADTVProgramGuideQueryItf self, 
  XAuint32 contentID, 
  XAuint32 timeAttribute, 
  XAtime * pTimeValue
);
```

#### Description

Gets the value of the given time attribute on the given content. If the given `timeAttribute` is not supported by the implementation, `XA_RESULT_FEATURE_UNSUPPORTED` is returned.

#### Pre-conditions

None.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>contentID</code></td>
<td>[in]</td>
<td>The content ID.</td>
</tr>
<tr>
<td><code>pTimeValue</code></td>
<td>[out]</td>
<td>Value of the given <code>timeAttribute</code>.</td>
</tr>
</tbody>
</table>

#### Return value

The return value can be one of the following:

- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_RESULT_FEATURE_UNSUPPORTED`

#### Comments

- 

### GetPurchaseBundlesOfService

```c
XAResult (*GetPurchaseBundlesOfService) ( 
  XADTVProgramGuideQueryItf self, 
  XAuint32 serviceID, 
  XAuint32 * pNumberOfBundles, 
  XAuint32 * pBundles
);
```

#### Description

Gets the purchase bundles to which the given service belongs.

#### Pre-conditions

None.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>serviceID</code></td>
<td>[in]</td>
<td>The service ID.</td>
</tr>
<tr>
<td><code>pNumberOfBundles</code></td>
<td>[in/out]</td>
<td>As an input, specifies the length of the <code>pBundles</code> array. As an output, specifies the returned number of bundles. Zero, if the given service does not belong to any bundle.</td>
</tr>
<tr>
<td><code>pBundles</code></td>
<td>[out]</td>
<td>IDs of the bundles.</td>
</tr>
</tbody>
</table>
### GetPurchaseBundlesOfService

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_BUFFER_INSUFFICIENT</td>
</tr>
</tbody>
</table>

| Comments | The caller of this method is responsible for allocating the pBundles array. |

### GetPurchaseBundlesOfContent

```c
XAResult (*GetPurchaseBundlesOfContent) ( 
    XADTVProgramGuideQueryItf self, 
    XAuint32 contentID, 
    XAuint32 * pNumberOfBundles, 
    XAuint32 * pBundles 
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Gets the purchase bundles where the given content belongs to.</th>
</tr>
</thead>
</table>

| Pre-conditions | None. |

<table>
<thead>
<tr>
<th>Parameters</th>
<th>********</th>
<th>**********</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentID</td>
<td>[in]</td>
<td>The content ID.</td>
</tr>
<tr>
<td>pNumberOfBundles</td>
<td>[in/out]</td>
<td>As an input, specifies the length of the pBundles array. As an output, specifies the returned number of bundles. Zero, if the given content does not belong to any bundle.</td>
</tr>
<tr>
<td>pBundles</td>
<td>[out]</td>
<td>IDs of the bundles.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_BUFFER_INSUFFICIENT</td>
</tr>
</tbody>
</table>

| Comments | The caller of this method is responsible for allocating the pBundles array. |
### GetAllPurchaseBundles

```c
XAResult (*GetAllPurchaseBundles) (  
    XADTVProgramGuideQueryItf self,  
    XAuint32 * pNumberOfBundles,  
    XAuint32 * pBundles  
    )
```

**Description**
Gets all the available purchase bundles listed in the program guide.

**Pre-conditions**
None.

**Parameters**
- `pNumberOfBundles` [in/out] As an input, specifies the length of the `pBundles` array. As an output, specifies the returned number of bundles. Zero, if the given content does not belong to any bundle.
- `pBundles` [out] IDs of the bundles.

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_RESULT_BUFFER_INSUFFICIENT`

**Comments**
The caller of this method is responsible for allocating the `pBundles` array.

### GetServicesOfPurchaseBundle

```c
XAResult (*GetServicesOfPurchaseBundle) (  
    XADTVProgramGuideQueryItf self,  
    XAuint32 bundleID,  
    XAuint32 * pNumberOfServices,  
    XAuint32 * pServices  
    )
```

**Description**
Gets the services included in the specified purchase bundle.

**Pre-conditions**
None.

**Parameters**
- `bundleID` [in] The bundle ID.
- `pNumberOfServices` [in/out] As an input, specifies the length of the `pServices` array. As an output, specifies the returned number of services. Zero, if the given bundle does not include any services.
- `pServices` [out] IDs of the services.
### GetServicesOfPurchaseBundle

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_BUFFER_INSUFFICIENT

**Comments**
The caller of this method is responsible for allocating the pServices array.

### GetContentsOfPurchaseBundle

```
XAresult (*GetContentsOfPurchaseBundle) (  
    XADTVProgramGuideQueryItf self,  
    XAuint32 bundleID,  
    XAuint32 * pNumberOfContents,
    XAuint32 * pContents  
);
```

**Description**
Gets the content IDs that are included in the given purchase bundle.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bundleID</td>
<td>[in]</td>
<td>The bundle ID.</td>
</tr>
<tr>
<td>pNumberOfContents</td>
<td>[in/out]</td>
<td>As an input, specifies the length of the pContents array. As an output,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specifies the returned number of content IDs. Zero, if the given bundle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>does not include any content IDs.</td>
</tr>
<tr>
<td>pContents</td>
<td>[out]</td>
<td>IDs of the contents.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_BUFFER_INSUFFICIENT

**Comments**
The caller of this method is responsible for allocating the pContents array.
### GetSubscribedPurchaseBundles

```
XAresult (*GetSubscribedPurchaseBundles) ( 
    XADTVProgramGuideQueryItf self, 
    XAuint32 * pNumberOfBundles, 
    XAuint32 * pBundles 
);
```

**Description**
Gets the currently subscribed purchase bundles.

**Preconditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pNumberOfBundles</td>
<td>[in/out]</td>
<td>As an input, specifies the length of the pBundles array. As an output, specifies the returned number of bundle IDs. Zero, if there are currently no subscriptions.</td>
</tr>
<tr>
<td>pBundles</td>
<td>[out]</td>
<td>IDs of the bundles.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_BUFFER_INSUFFICIENT

**Comments**
The caller of this method is responsible for allocating the pBundles array.

### GetBundleStringAttrib

```
XAresult (*GetBundleStringAttrib) ( 
    XADTVProgramGuideQueryItf self, 
    XAuint32 bundleID, 
    XAuint32 stringAttribute, 
    XAuint32 * pStringLength, 
    XAchar * pStringValue 
);
```

**Description**
Gets the value of the given string attribute on the given purchase bundle. If the given stringAttribute is not supported by the implementation, XA_RESULT_FEATURE_UNSUPPORTED is returned.

**Preconditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bundleID</td>
<td>[in]</td>
<td>The bundle ID.</td>
</tr>
<tr>
<td>stringAttribute</td>
<td>[in]</td>
<td>See XA_DTV_EPG_ATTR_STR_macros..</td>
</tr>
<tr>
<td>pStringLength</td>
<td>[in/out]</td>
<td>As an input, specifies the length of the pStringValue array. As an output, specifies the length of the returned string. May be zero if not sent.</td>
</tr>
<tr>
<td>pStringValue</td>
<td>[out]</td>
<td>Value of the given stringAttribute</td>
</tr>
</tbody>
</table>
### GetBundleStringAttrib

<table>
<thead>
<tr>
<th><strong>Return value</strong></th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_FEATURE_UNSUPPORTED</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_BUFFER_INSUFFICIENT</td>
</tr>
</tbody>
</table>

| **Comments** | The caller of this method is responsible for allocating the pStringValue array. |

### GetBundleIntAttrib

```c
XAResult (*GetBundleIntAttrib) (  
    XADTVProgramGuideQueryItf self,  
    XAuint32 bundleID,  
    XAuint32 intAttribute,  
    XAuint32 * pValue  
);  
```

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Gets the value of the given integer attribute on the given purchase bundle. If the given intAttribute is not supported by the implementation, XA_RESULT_FEATURE_UNSUPPORTED is returned.</th>
</tr>
</thead>
</table>

| **Pre-conditions** | None. |

<table>
<thead>
<tr>
<th><strong>Parameters</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bundleID</td>
<td>[in]</td>
<td>The bundle ID.</td>
</tr>
<tr>
<td>pValue</td>
<td>[out]</td>
<td>Value of the given intAttribute</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Return value</strong></th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_FEATURE_UNSUPPORTED</td>
</tr>
</tbody>
</table>

**Comments** | |
### GetBundleTimeAttrib

```c
XAResult (*GetBundleTimeAttrib) (
    XADTVProgramGuideQueryItf self,
    XAuint32 bundleID,
    XAuint32 timeAttribute,
    XAtime * pTimeValue
);
```

**Description**
Gets the value of the given time attribute on the given purchase bundle. If the given `timeAttribute` is not supported by the implementation, `XA_RESULT_FEATURE_UNSUPPORTED` is returned.

**Preconditions**
None.

**Parameters**
- `bundleID` [in]: The bundle ID.
- `timeAttribute` [in]: See `XA_DTV_EPG_ATTR_TIME_` macros.
- `pTimeValue` [out]: Value of the given `timeAttribute`.

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_RESULT_FEATURE_UNSUPPORTED`

**Comments**
-

### CreateServiceConnectionInfo

```c
XAResult (*CreateServiceConnectionInfo) (
    XADTVProgramGuideQueryItf self,
    XAuint32 IDType,
    XAuint32 ID,
    void * pInternetConnection,
    XADTVServiceConnectionInfo * pServiceConnectionInfo
);
```

**Description**
Collects and outputs service connection information to be used with the `XADTVServiceInputSelectorItf::ConnectUsingServiceGuideData` method on the service. As input this method takes a content- or service ID, collects the necessary connection data and outputs the data in the supplied `pServiceConnectionInfo` struct.

**Preconditions**
None.

**Parameters**
- `IDType` [in]: Indicates the type of ID supplied. See the `XA_DTV_EPG_CONTENT_ID_TYPE` macros for more information.
- `ID` [in]: The ID for the content to generate the connection info from. This ID cannot be a bundle ID.
### CreateServiceConnectionInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pInternetConnection</td>
<td>[in] This parameter is used for unicast or multicast connections that require an internet connection. If NULL is used the implementation will select a connection automatically. For technologies not requiring an internet connection this parameter is ignored. See the XADTVSourceUtilitiesItf::QueryInternetConnections method for more information on retrieving system internet connections.</td>
</tr>
<tr>
<td>pServiceConnectionInfo</td>
<td>[out] The generated connection info containing data to be used to configure the DTVSource and Service objects.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

- 

**See Also**

XADTVSourceUtilitiesItf::QueryInternetConnections
7.6. XADTVProgramGuideUpdateItf

Description

This interface is the main interface for the program guide. It is responsible for initialization of the program guide and notification of changes.

The interface also has methods to get and set the current state of the program guide to and from the file system. This functionality can be used to let the application supply the EPG state between sessions since retrieval of a full EPG might take a long time.

This interface is a mandated interface on the Program Guide Manager (See section 6.3) object.
Prototype

XA_API extern const XAInterfaceID XA_IID_DTVPROGRAMGUIDEUPDATE;

struct XADTVProgramGuideUpdateItf;
typedef const struct XADTVProgramGuideUpdateItf * const * XADTVProgramGuideUpdateItf;

struct XADTVProgramGuideUpdateItf_ {
   XAResult (*EnableEPGListening) (
      XADTVProgramGuideUpdateItf self,
      XAuint32 bearerID,
      XAboolean IncludeEPGData,
      void * pContext
   );
   XAResult (*DisableEPGListening) (
      XADTVProgramGuideUpdateItf self,
      XAuint32 bearerID
   );
   XAResult (*SetEPG) (
      XADTVProgramGuideUpdateItf self,
      XAuint32 bearerID,
      XAchar * pURI
   );
   XAResult (*GetEPG) (
      XADTVProgramGuideUpdateItf self,
      XAuint32 BearerID,
      XAchar * pURI,
      void * pContext
   );
   XAResult (*IsEPGListeningEnabled) (
      XADTVProgramGuideUpdateItf self,
      XAuint32 BearerID,
      XAboolean * pIsListening
   );
   XAResult (*IsEPGStoragePersistent) (
      XADTVProgramGuideUpdateItf self,
      XAboolean * pIsPersistent
   );
   XAResult (*RegisterEPGGetEPGCallback) (
      XADTVProgramGuideUpdateItf self,
      xaEPGGetEPGCallback Callback,
      void * pContext
   );
   XAResult (*RegisterEPGUpdateCallback) (
      XADTVProgramGuideUpdateItf self,
      xaEPGUpdateCallback Callback,
      void * pContext
   );
};

Interface ID

e00aa3f0-ba4f-11de-8a39-0800200c9a66
**Callbacks**

**xaEPGUpdateCallback**

```c
typedef void (XAAPIENTRY * xaEPGUpdateCallback)(
    XADTVProgramGuideUpdateItf Caller,
    XAuint32 bearerID,
    XAchar * pEPGName,
    XAchar * pEPGData,
    void * pContext
);
```

**Description**
Notifies the application that a new EPG fragment has arrived.

If the `IncludeEPGData` boolean parameter is set to `TRUE` when `StartEPGListing` was called this callback also returns the actual unmodified EPG data.

This callback is triggered each time a new EPG fragment arrives in the stream.

**Parameters**
- **Caller** [in] Interface for which this callback was registered.
- **bearerID** [in] Identifies the bearer for which the EPG was updated.
- **pEPGName** [in] A pointer to the NULL-terminated string containing the identifying name of the EPG that has been updated.
- **pEPGData** [in] A pointer to the unmodified, raw EPG data as it was delivered. This parameter may be NULL if the `IncludeEPGData` parameter was set to `FALSE` when `StartEPGListing` was called.
- **pContext** [in] User context data that is supplied when the callback method is registered.

**Comments**
If multiple bearers are associated with the current Program Guide Manager, the bearerID parameter is used to identify the bearer to which the Program Guide fragment belongs.

**See Also**
- `XADTVProgramGuideUpdateItf::StartEPGListing()`

**xaEPGGetEPGCallback**

```c
typedef void (XAAPIENTRY * xaEPGGetEPGCallback)(
    XADTVProgramGuideUpdateItf Caller,
    XAuint32 bearerID,
    void * pContext
);
```

**Description**
Notifies the application that the get of an EPG is complete.

**Parameters**
- **Caller** [in] Interface for which this callback was registered.
- **bearerID** [in] Identifies the bearer on which the update was received.
- **pContext** [in] User context data that is supplied when the callback method is registered.

**Comments**

**See Also**
- `GetEPG()`
### Methods

#### EnableEPGListening

| XAResult (*EnableEPGListening) ( |
| XADTVProgramGuideUpdateItf self, |
| XAuint32 bearerID, |
| XAboolean IncludeEPGData, |
| void * pContext |
| ); |

**Description**

Starts the listening for EPG data using a specified tuned bearer identified with the `bearerID` supplied.

It is possible to associate multiple bearers to a single Program Guide Manager object. When multiple bearers are associated with the Program Guide Manager, each bearer associated is identified by its unique `bearerID`.

**Pre-conditions**

The `bearerID` supplied must identify a previously tuned bearer. See the `XADTVSourceBroadcastItf::SetScanInfo` for information about tuning a broadcast bearer. The EPG object is not limited to handling broadcast bearers.

**Parameters**

| bearerID | [in] | The ID of the previously tuned bearer. |
| IncludeEPGData | [in] | Indicates whether the callback is to return the raw EPG data fragment to the application on every callback. |
| pContext | [in] | User context data that is supplied when the callback method is registered. |

**Return value**

The return value can be one of the following:

- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

**Comments**

See Also

XADTVSourceBroadcastItf::SetScanInfo()
### DisableEPGListing

```c
XAResult (*DisableEPGListing) (  
  XADTVProgramGuideUpdateItf self,  
  XAuint32 bearerID  
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Stops the previously started listening for EPG data associated with a specified <code>bearerID</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>The <code>bearerID</code> specified must identify a bearer on which <code>StartEPGListening</code> previously has been called to have any effect. If <code>StartEPGListening</code> was not previously called this method will return <code>XA_RESULT_PARAMETER_INVALID</code>.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td><code>- bearerID</code></td>
<td>[in] The ID of the listening bearer.</td>
</tr>
<tr>
<td>Return value</td>
<td>The return value can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td>Comments</td>
<td></td>
</tr>
<tr>
<td>See Also</td>
<td><code>StartEPGListening()</code></td>
</tr>
</tbody>
</table>

### SetEPG

```c
XAResult (*SetEPG) (  
  XADTVProgramGuideUpdateItf self,  
  XAuint32 bearerID,  
  XAchar * pURI  
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets an Electronic Program Guide associated with a specified <code>bearerID</code>. The <code>pURI</code> parameter points to an actual Electronic Program Guide to set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>None.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td><code>- pURI</code></td>
<td>[in] A pointer to a <code>NULL</code>-terminated string containing a URI that identifies an existing EPG.</td>
</tr>
</tbody>
</table>
### SetEPG

<table>
<thead>
<tr>
<th><strong>Return value</strong></th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| **Comments** | The EPG identified by the `pURI` parameter could be either a previously saved EPG retrieved by GetEPG or an EPG from an external source. For example a XML based EPG residing on a web server. |

| **See Also** | GetEPG() |

### GetEPG

```c
XAResult (*GetEPG) ( 
    XADTVProgramGuideUpdateItf self, 
    XAuint32 bearerID, 
    XAchar * pURI, 
    xaEPGGetEPGCallback Callback, 
    void * pContext)
```

| **Description** | Gets an EPG associated with a specified bearerID from the implementation. The location where to save the EPG is identified by the URI supplied. The EPG returned after a call to this method is not required to be parseable by the application. The most common purpose of this method is to make it possible for an application to retain the EPG state between sessions. This method is especially handy if the implementation does not support persistence of its EPG data between sessions. The only required use of the data returned is to use it as input to the SetEPG method. This method is asynchronous and when the save is complete the callback function supplied is called. |

| **Pre-conditions** | An existing EPG. |

<table>
<thead>
<tr>
<th><strong>Parameters</strong></th>
<th>** selves</th>
<th>** bearerID</th>
<th>** pURI</th>
<th>** pContext</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>The ID of the tuned bearer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pURI</td>
<td>[in]</td>
<td>A pointer to a NULL-terminated string containing a URI identifying a location where to save the EPG.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pContext</td>
<td>[in]</td>
<td>User context data that is supplied when the callback method is registered.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## GetEPG

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

### Comments

See Also

SetEPG()

## IsEPGListeningEnabled

```c
XAResult (*IsEPGListeningEnabled) (XADTVProgramGuideUpdateItf self, XAuint32 BearerID, XAboolean * pIsListening);
```

### Description

Queries the specified bearer to see whether it is currently listening for EPG data.

### Preconditions

None.

### Parameters

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BearerID</td>
<td>[in]</td>
<td>The ID of the bearer to query.</td>
</tr>
<tr>
<td>pIsListening</td>
<td>[out]</td>
<td>The listening state of the bearer.</td>
</tr>
</tbody>
</table>

### Return value

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

### Comments

See Also

EnableEPGListening(), DisableEPGListening()
### IsEPGStoragePersistent

**Description**
Queries to find out if it supports saving the EPG data persistently between sessions. If the implementation does not support persistence of the EPG data between sessions and the application does not want to re-download the EPG, the application must use the GetEPG and SetEPG methods to retain the EPG data between runs.

**Pre-conditions**
None.

**Parameters**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pIsPersistent</td>
<td>[out]</td>
<td>Indicates whether the implementation supports persistence of the EPG data between runs.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
None.

**See Also**
SetEPG(), GetEPG()

---

### RegisterEPGUpdateCallback

**Description**
Sets the callback method that is to be triggered each time a new EPG fragment arrives.

**Pre-conditions**
None.

**Parameters**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback</td>
<td>[in]</td>
<td>Specifies the callback method.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in]</td>
<td>User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
None.

**See Also**
xaEPGUpdateCallback()
**RegisterEPGGetEPGCallback**

```c
XAResult (*RegisterEPGGetEPGCallback) ( 
    XADTVProgramGuideUpdateItf self, 
    xaEPGGetEPGCallback Callback, 
    void * pContext 
);
```

**Description**  
The callback method to be called when the `GetEPG` save is completed.

**Pre-conditions**  
None

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback</td>
<td>[in]</td>
<td>Specifies the callback method.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in]</td>
<td>User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Return value**  
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

**Comments**  
None

**See Also**  
`xaEPGGetEPGCallback()`
7.7. XADTVRecorderModeItf

Description

XADTVRecorderModeItf is used to record the streams from a Digital TV service. This interface is a recorder object extension interface. Streams to record are chosen with help of the select streams mechanism in the stream information interface. This interface initializes the configurable recording parameters specific for the Digital TV Extension.

This interface is a mandated interface on the Recorder (See section 6.4) object.

Prototype

XA_API extern const XAInterfaceID XA_IID_DTVRECORDERMODE;

struct XADTVRecorderModeItf_;
typedef const struct XADTVRecorderModeItf_ * const *
XADTVRecorderModeItf;
struct XADTVRecorderModeItf_ {
    XAresult (*QueryRecordModeCapabilities) (XADTVRecorderModeItf self, XAuint32 * pRecordModeCapabilities);
    XAresult (*SetDTVRecordMode) (XADTVRecorderModeItf self, XAuint32 RecordMode);
    XAresult (*GetDTVRecordMode) (XADTVRecorderModeItf self, XAuint32 * pRecordMode);
};

Interface ID

7fac2c90-c463-11de-8a39-0800200c9a66

Callbacks

None
### Methods

#### QueryRecordModeCapabilities

```c
XAResult (*QueryRecordModeCapabilities) (XADTVRecorderModeItf self, XAuint32 * pRecordModeCapabilities);
```

**Description**

Gets information about the capability of the device. Returns whether the implementation is capable of different record modes.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pRecordModeCapabilities</td>
<td>[out]</td>
<td>The capabilities of the device. Defined as a bitmask with implementation specific record mode capabilities. See the XA_DTV_RECORDER_MODE macro definition.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

See Also

SetDTVRecordMode

#### SetDTVRecordMode

```c
XAResult (*SetDTVRecordMode) (XADTVRecorderModeItf self, XAuint32 RecordMode);
```

**Description**

Tells the implementation how to record the digital TV service data.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordMode</td>
<td>[in]</td>
<td>The record mode as described by the capabilities of the implementation by a call to QueryRecordModeCapabilities.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

See Also

QueryRecordModeCapabilities
GetDTVRecordMode

XAResult (*GetDTVRecordMode) (XADTVRecorderModeItf self,
XAuint32 * pRecordMode);

**Description**

Gets the current record mode.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pRecordMode</td>
<td>[out]</td>
<td>The current recording mode.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

See Also

SetDTVRecordMode()


7.8. XADTVServiceDataDeliveryItf

Description

XADTVServiceDataDeliveryItf is used to handle data delivery associated with the service.

For the file delivery to work, the service must be connected to a data source using the XADTVServiceInputSelectorItf prior to calling the methods in this interface.

It is up to the implementation to map the functionality of this interface to the underlying delivery mechanisms of the connected bearers that provide file delivery. This interface functions the same with respect to the applications regardless of how the bearer delivers data. If the data carousel is delivered in an IP-layer within a multiplex on a bearer, the underlying implementation maps the delivery mechanism to this interface.

This interface is an optional interface on the Service (See section 6.5) object.
Prototype

XA_API extern const XAInterfaceID XA_IID_DTVSERVICEDATADELIVERY;

struct XADTVServiceDataDeliveryItf_;
typedef const struct XADTVServiceDataDeliveryItf_ * const *
XDADTVServiceDataDeliveryIf;
struct XADTVServiceDataDeliveryItf_ {
  XAresult (*setEnabled) (  
    XADTVServiceDataDeliveryItf self,  
    XAboolean Enabled,  
    void * Context
  );
  XAresult (*isEnabled) (  
    XADTVServiceDataDeliveryItf self,  
    XAboolean * pIsEnabled
  );
  XAresult (*setFileDeliveryCache) (  
    XADTVServiceDataDeliveryItf self,  
    XAuint32 Size,  
    XAchar * pUri
  );
  XAresult (*list) (  
    XADTVServiceDataDeliveryItf self,  
    XAuint32 * NumberOfFiles,  
    XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor,  
    XAboolean * pIsCached
  );
  XAresult (*download) (  
    XADTVServiceDataDeliveryItf self,  
    XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor,  
    XAchar * pUri,  
    void * Context
  );
  XAresult (*abortDownload) (  
    XADTVServiceDataDeliveryItf self,  
    XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor
  );
  XAresult (*registerServiceFileDescriptionCallback) (  
    XADTVServiceDataDeliveryItf self,  
    xaServiceFileDescriptionUpdateCallback Callback,  
    void * pContext
  );
  XAresult (*registerServiceFileDownloadProgressCallback) (  
    XADTVServiceDataDeliveryItf self,  
    xaDTVServiceFileDownloadProgressCallback Callback,  
    void * pContext
  );
};

Interface ID

4d04cc20-c463-11de-8a39-0800200c9a66
## Callbacks

### xaServiceFileDescriptionUpdateCallback

```c
typedef void (XAAPIENTRY * xaServiceFileDescriptionUpdateCallback) (
    XADTVServiceDataDeliveryItf Caller,
    XAuint32 UpdateType,
    XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor,
    void * pContext
);
```

**Description**

Notifies the application that a new file descriptor update has arrived on the service.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller</td>
<td>Interface on which this callback was registered.</td>
</tr>
<tr>
<td>UpdateType</td>
<td>Indicates if this filedescriptor received is a new, updated or removed instance. Refer to the XA_DTV_SERVICE_FILE_DESCRIPTION macros for more information.</td>
</tr>
<tr>
<td>pFileDescriptor</td>
<td>The file descriptor.</td>
</tr>
<tr>
<td>pContext</td>
<td>User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Comments**

See Also

### xaDTVServiceFileDownloadProgressCallback

```c
typedef void (XAAPIENTRY * xaDTVServiceFileDownloadProgressCallback) (
    XADTVServiceDataDeliveryItf Caller,
    XAuint32 ProgressStatus,
    XApermille Progress,
    XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor,
    void * pContext
);
```

**Description**

Notifies the application about file download progress.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller</td>
<td>Interface on which this callback was registered.</td>
</tr>
<tr>
<td>ProgressStatus</td>
<td>Indicates if the file is still downloading. Refer to the XA_DTV_SERVICE_FILE_DL_PROGRESS macros for more information.</td>
</tr>
<tr>
<td>Progress</td>
<td>The amount of the file downloaded so far.</td>
</tr>
<tr>
<td>pFileDescriptor</td>
<td>The file descriptor associated with this progress.</td>
</tr>
<tr>
<td>pContext</td>
<td>User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Comments**

See Also

## Methods

### SetEnabled
### SetEnabled

```c
XAResult (*SetEnabled)(
    XADTVServiceDataDeliveryItf self,
    XABool Enabled,
    void * Context
);
```

**Description**
Enables or disables the file delivery mechanism and start listening for FDT (File Delivery Table) entries and similar. When a file delivery entry descriptor arrives the application is notified about it by a callback notification `xaDTVSERVICEFILEDESCRIPTIONUPDATECALLBACK` if the callback has been registered.

**Pre-conditions**
None.

**Parameters**
- **Enabled** [in] `TRUE` if the service should be enabled, `FALSE` if the service should be disabled.
- **Context** [in] User context data that is to be returned as part of the callback method.

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

**Comments**

**See Also**

### IsEnabled

```c
XAResult (*SetEnabled)(
    XADTVServiceDataDeliveryItf self,
    XABool * pIsEnabled,
    void * Context
);
```

**Description**
Queries to find out if the file delivery mechanism is enabled.

**Pre-conditions**
None.

**Parameters**
- **pIsEnabled** [out] Indicates whether the file delivery mechanism is enabled or not.
- **Context** [in] User context data that is to be returned as part of the callback method.

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

**Comments**

**See Also**
### SetFileDeliveryCache

**XAResult (*SetFileDeliveryCache) (**

- XADTVServiceDataDeliveryItf self,
- XAuint32 Size,
- XAchar * pUri

**);**

**Description**
Specifies a maximum allowed file cache for the file delivery mechanism. If a size and a valid writable URI are set, the implementation automatically saves all received files in a file cache.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>[in]</td>
<td>The maximum file system size that the file cache may use.</td>
</tr>
<tr>
<td>pUri</td>
<td>[in]</td>
<td>A NULL-terminated string containing a URI to where to store the cached files.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

**See Also**

### List

**XAResult (*List) (**

- XADTVServiceDataDeliveryItf self,
- XAuint32 * NumberOfFiles,
- XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor,
- XAboolean * pIsCached

**);**

**Description**
Retrieves an array of all currently known File Descriptors available for this service. Note that this list is not constant and directly after connecting a service this list will be 0. It depends heavily on the DTVSource provider. The ppFileDescriptor and the pIsCached arrays are synchronized. The pIsCached array indicates whether a file described in the ppFileDescriptor is already downloaded and available in the cache.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
### List

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfFiles</td>
<td>[in/out]</td>
<td>As in-parameter this parameter indicates how large array of pFileDescriptor and pIsCached entries that are allocated. As out-parameter this parameter tells how many entries that were filled in the pFileDescriptor and the pIsCached arrays.</td>
</tr>
<tr>
<td>pFileDescriptor</td>
<td>[out]</td>
<td>This parameter contains the file descriptors currently available in the service.</td>
</tr>
<tr>
<td>pIsCached</td>
<td>[out]</td>
<td>This parameter contains an indication if the files are cached.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_BUFFER_INSUFFICIENT

**Comments**

The caller of this method is responsible for allocating the pFileDescriptor and pIsCached arrays.

### Download

```c
XAresult (*Download) (XADTVServiceDataDeliveryItf self, XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor, XAchar * pUri, void * Context);
```

**Description**

Requests a download of a specified file.

The download progress is notified to the application using the xaDTVServiceFileDownloadProgressCallback callback.

**Preconditions**

None.

**Parameters**

- **pFileDescriptor** [in] The file descriptor associated with the file to download.
- **pUri** [in] A NULL-terminated string containing the URI to where to save the file.
- **Callback** [in] The callback method to receive the progress notifications.
- **Context** [in] User context data that is to be returned as part of the callback method.

**Return value**

The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
### AbortDownload

```c
XAresult (*AbortDownload) (  
    XADTVServiceDataDeliveryItf self,  
    XADTVServiceDataDeliveryFileDescriptor * pFileDescriptor  
);
```

**Description**
Aborts a current download in progress. After this method is successfully called, a final notification is sent to the application thru the `xaDTVServiceFileDownloadProgressCallback` callback indicating that the download was aborted.

**Pre-conditions**
None.

**Parameters**
- `pFileDescriptor` [in] The file descriptor associated with the file to abort the ongoing download.

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

**Comments**

**See Also**
`xaDTVServiceFileDownloadProgressCallback`

### RegisterServiceFileDescriptionCallback

```c
XAresult (*RegisterServiceFileDescriptionCallback) (  
    XADTVServiceDataDeliveryItf self,  
    xaDTVServiceFileDescriptionCallback Callback,  
    void * pContext  
);
```

**Description**
Sets the callback for service file description event notifications.

**Pre-conditions**
None.

**Parameters**
- `Callback` [in] Specifies the callback method.
- `pContext` [in] User context data that is supplied when the callback method is registered.

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`

**Comments**

**See Also**
`xaDTVServiceFileDescriptionCallback()`
### RegisterServiceFileDownloadProgressCallback

**XAResult (*RegisterServiceFileDownloadProgressCallback) (**
* XADTVServiceDataDeliveryItf self, *
* xaDTVSserviceFileDownloadProgressCallback Callback, *
* void * pContext *
* );
**

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets the callback method to receive the progress notifications.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pre-conditions</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td><strong>self</strong> [in] Interface self-reference.</td>
</tr>
<tr>
<td></td>
<td><strong>Callback</strong> [in] Specifies the callback method.</td>
</tr>
<tr>
<td></td>
<td><strong>pContext</strong> [in] User context data that is supplied when the callback method is registered.</td>
</tr>
<tr>
<td><strong>Return value</strong></td>
<td>The return value can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>See Also</strong></td>
<td>xaDTVSserviceFileDownloadProgressCallback()</td>
</tr>
</tbody>
</table>
7.9. XADTVServiceInputSelectorItf

Description

XADTVServiceInputSelectorItf is used to select service source. The service input selector controls from what entity and how the service gets its data. Once connected the connection cannot be changed.

This interface is a mandated interface on the Service (See section 6.5) object.

Prototype

```
XA_API extern const XAInterfaceID XA_IID_DTVSERVICEINPUTSELECTOR;

struct XADTVServiceInputSelectorItf_;
typedef const struct XADTVServiceInputSelectorItf_ * const *
    XADTVServiceInputSelectorItf;
struct XADTVServiceInputSelectorItf_ {
    XAresult (*ConnectUsingProgramGuideData) (  
        XADTVServiceInputSelectorItf self,  
        XADTVServiceConnectionInfo * pServiceConnectionInfo
    );
    XAresult (*ConnectUsingBearerID) (  
        XADTVServiceInputSelectorItf self,  
        XAuint32 bearerID
    );
    XAResult (*IsConnected) (  
        XADTVServiceInputSelectorItf self,  
        XAboolean * pIsConnected
    );
};
```

Interface ID

a29b99b0-ba50-11de-8a39-0800200c9a66

Callbacks

```
xaDTVServiceConnectionLostCallback

typedef void (XAAPIENTRY * xaDTVServiceConnectionLostCallback) (  
    XADTVServiceInputSelectorItf Caller,  
    void * pContext
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Notifies the application that a connection was lost.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td><strong>Caller</strong> [in] Interface on which this callback was registered.</td>
</tr>
<tr>
<td></td>
<td><strong>pContext</strong> [in] User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

See Also
## Methods

### ConnectUsingProgramGuideData

```c
XAResult (*ConnectUsingProgramGuideData) (XADTVServiceInputSelectorItf self,
XADTVServiceConnectionInfo * pServiceConnectionInfo);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Connects the service using data queried from the program guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>None.</td>
</tr>
<tr>
<td>Return value</td>
<td>The return value can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td>Comments</td>
<td></td>
</tr>
</tbody>
</table>

### ConnectUsingBearerID

```c
XAResult (*ConnectUsingBearerID) (XADTVServiceInputSelectorItf self,
XAuint32 bearerID);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Connects the service using an already tuned bearer from the DTVSource object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>The specified bearer must already be tuned to a specific service.</td>
</tr>
<tr>
<td></td>
<td>Bearers that do not support pointing to a single service without using a program guide object cannot use this method to receive data.</td>
</tr>
<tr>
<td>Return value</td>
<td>The return value can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td>Comments</td>
<td></td>
</tr>
<tr>
<td>See Also</td>
<td>XADTVSourceBroadcastItf::SetScanInfo</td>
</tr>
<tr>
<td></td>
<td>XADTVSourceBroadcastItf::GetScanInfo</td>
</tr>
</tbody>
</table>
### IsConnected

```c
XAResult (*IsConnected) (
    XADTVServiceInputSelectorItf self,
    XAboolean * pIsConnected
);
```

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Queries to find out if it is connected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pre-conditions</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Parameters</strong></th>
<th><strong>[in]</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td></td>
<td>Interface self-reference.</td>
</tr>
<tr>
<td>pIsConnected</td>
<td><strong>[out]</strong></td>
<td>Indicates whether the service is connected or not.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Return value</strong></th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| **Comments**     |                                               |
|------------------|                                               |
| **See Also**     |                                               |
7.10. XADTVSourceBroadcastItf

Description

XADTVSourceBroadcastItf is used to scan for broadcasts within a bearer.

This interface is a mandated interface on the DTVSource object (See section 4.1.1). If one bearer technology is supported all other bearer technology interfaces (XADTVSourceBroadcastItf, XADTVSourceLocalItf, XADTVSourceMulticastItf and XADTVSourceUnicastItf) are optional to support.

Scanning for services is accomplished by the following steps (also shown in Figure 14 below):

1. Find out the bearer ID of the broadcast bearer you want to scan by using the XADTVSourceUtilitiesItf::GetStaticBearers method.
2. Check the state of that bearer and, if needed, enable the bearer by changing its power state to XA_DTV_BEARER_STATE_ON by using the XADTVSourceUtilitiesItf::SetBearerState method.
3. Register the scanning callback function by calling the XADTVSourceBroadcastItf::RegisterScanningCallback method.
4. Start the scanning by calling the XADTVSourceBroadcastItf::EnableScan(TRUE) method.
5. For each service or multiplex found, the callback registered in step 3 will be called by the implementation with information about the found service or multiplex. It is up to the application to save the found services or multiplexes to be used later.
6. The scanning may be cancelled by the application at any time by calling the XADTVSourceBroadcastItf::EnableScan(FALSE) method. The implementation will automatically cancel the scanning when there are no more frequencies available to scan. This indicated by a last call to the callback with data containing a NULL value for the service.

The found services or multiplexes are later used by the application to tune the bearer to that specific service or multiplex.
Figure 14: Scanning for Broadcast Services
Prototype

XA_API extern const XAInterfaceID XA_ID_DTVSOURCEBROADCAST;

struct XADTVSourceBroadcastItf_;
typedef const struct XADTVSourceBroadcastItf_ * const *
XADTVSourceBroadcastItf;

struct XADTVSourceBroadcastItf_ {
    XAresult (*EnableScan) (  
        XADTVSourceBroadcastItf self,  
        XAboolean enable,  
        XAuint32 bearerID,  
        void * pContext
    );

    XAresult (*SetScanInfo) (  
        XADTVSourceBroadcastItf self,  
        XAuint32 bearerID,  
        XADTVSourceScanInfo * pScanInfo
    );

    XAresult (*GetScanInfo) (  
        XADTVSourceBroadcastItf self,  
        XAuint32 bearerID,  
        XADTVSourceScanInfo * pScanInfo
    );

    XAresult (*IsScanning) (  
        XADTVSourceBroadcastItf self,  
        XAuint32 bearerID,  
        XAboolean * pIsScanning
    );

    XAresult (*RegisterSourceScanningCallback) (  
        XADTVSourceBroadcastItf self,  
        xADTVSourceScanningCallback Callback,  
        void * pContext
    );
};

Interface ID

213f8170-c463-11de-8a39-0800200c9a66
Callbacks

**xaDTVSourceScanningCallback**

```c
typedef void (XAAPIENTRY * xaDTVSourceScanningCallback) (
    XADTVSourceBroadcastItf Caller,
    XAuint32 bearerID,
    XADTVSourceScanInfo * pScanInfo,
    void * pContext
);
```

**Description**

Notifies the application that a frequency, content or platform has been found.

One callback per found frequency, content or platform will be returned.

Scan is complete when `pScanInfo` is returned as NULL. The scan is automatically cancelled upon completion. If the scan is cancelled manually, a final callback with `pScanInfo` set to NULL will be issued upon termination of the scan.

The scan mechanism returns information for single services when possible.

For services requiring and EPG to retrieve the single service information, such as for a DVB-H platform, the scan information is returned as a complete set of multiple services.

**Parameters**

- **Caller** 
  [in] Interface on which this callback was registered.

- **bearerID** 
  [in] The ID of the bearer that is performing a scan.

- **pScanInfo** 
  [in] The returned scan specific data for each service or set of services. When possible, will contain information for each service in the multiplex, otherwise will contain information for the complete set of services.

- **pContext** 
  [in] User context data that is supplied when the callback method is registered.

**Comments**

**See Also**

**Methods**

**EnableScan**

```c
XAResult (*EnableScan) ( 
    XADTVSourceBroadcastItf self,
    XAboolean enable,
    XAuint32 bearerID,
    void * pContext
);
```

**Description**

Enables or disables a service scan on a specified bearer.

**Pre-conditions**

None.

**Parameters**

- **self** 
## EnableScan

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>[in]</td>
<td>Set to <strong>TRUE</strong> if the scan is to be enabled; set to <strong>FALSE</strong> if the scan is to be disabled.</td>
</tr>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>The callback method to call on each service found during the scan.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in]</td>
<td>User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

None

## SetScanInfo

```c
XAresult (*SetScanInfo) (  
    XADTVSourceBroadcastItf self,  
    XAuint32 bearerID,  
    XADTVSourceScanInfo * pScanInfo  
);
```

**Description**

Tunes the specified bearer to the supplied `ScanInfo`.

**Pre-conditions**

None.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>The bearer ID of the bearer to tune.</td>
</tr>
<tr>
<td>pScanInfo</td>
<td>[in]</td>
<td>The information returned from the scan. This data is used to tune the bearer to a specific service or multiple services.</td>
</tr>
</tbody>
</table>
### SetScanInfo

**Return value**

- **XA_RESULT_PERMISSION_DENIED** – The bearer may be locked to a specific provider or platform. Use the description of the bearer returned by the DTVSourceUtilitiesItf interface to see if the bearer is locked.
- **XA_RESULT_SUCCESS**
- **XA_RESULT_PARAMETER_INVALID**

**Comments**

**See Also**

GetScanInfo()

### GetScanInfo

```c
XAResult (*GetScanInfo) (  
    XADTVSourceBroadcastItf self,  
    XAuint32 bearerID,  
    XADTVSourceScanInfo * pScanInfo  
);
```

**Description**

Retrieves the currently selected platform or service for the bearer.

**Pre-conditions**

SetScanInfo must have been set before calling this method.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>The bearer ID of the bearer to retrieve the scan info from.</td>
</tr>
<tr>
<td>pScanInfo</td>
<td>[out]</td>
<td>The currently tuned platform or service.</td>
</tr>
</tbody>
</table>

**Return value**

- **XA_RESULT_PRECONDITIONS_VIOLATED** – No Scan Info has previously been set.
- **XA_RESULT_SUCCESS**

**Comments**

**See Also**

SetScanInfo()

### IsScanning

```c
XAResult (*IsScanning) (  
    XADTVSourceBroadcastItf self,  
    XAuint32 bearerID,  
    XABoolean * pIsScanning  
);
```

**Description**

Returns TRUE if there is an active service scan on a specific bearer.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>The ID of the bearer to query for ongoing scan.</td>
</tr>
</tbody>
</table>
### IsScanning

<table>
<thead>
<tr>
<th>pIsScanning</th>
<th>[out]</th>
<th>Indicates whether there is an active service scan on a specific bearer or not.</th>
</tr>
</thead>
</table>

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

<table>
<thead>
<tr>
<th>Return value</th>
<th>None</th>
</tr>
</thead>
</table>

| Comments | |

### RegisterSourceScanningCallback

```c
XAResult (*RegisterSourceScanningCallback) (  
    XADTVSourceBroadcastItf self,  
    xaADTVSourceScanningCallback Callback,  
    void * pContext  
);```

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets the callback for scanning event notifications.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback</td>
<td>[in] Specifies the callback method.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in] User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>- XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| Comments | None |

| See Also | xaADTVSourceScanningCallback() |
7.11. XADTVSourceLocalItf

**Description**

This interface represents bearers that rely on local data. An example of such a bearer is a file in the file system. This interface holds methods specific to the local family. This interface is a mandated interface on the DTVSource (See section 0) object. If one bearer technology is supported all other bearer technology interfaces (XADTVSourceBroadcastItf, XADTVSourceLocalItf, XADTVSourceMulticastItf and XADTVSourceUnicastItf) are optional to support.

**Prototype**

```c
XA_API extern const XAInterfaceID XA_IID_DTVSOURCELOCAL;

struct XADTVSourceLocalItf_;
typedef const struct XADTVSourceLocalItf * const * XADTVSourceLocalItf;

struct XADTVSourceLocalItf_ {
    XAResult (*Associate) (XADTVSourceLocalItf self, XADTVLocalAssociateInfo * pLocalAssociateInfo, XAuint32 * pBearerID);
    XAResult (*Disassociate) (XADTVSourceLocalItf self, XAuint32 bearerID);
    XAResult (*IsAssociated) (XADTVSourceLocalItf self, XAboolean * pIsAssociated);
};
```

**Interface ID**

36f7ca67-38bd-4db2-811d-bf063b543f36

**Callbacks**

None
### Methods

#### Associate

```c
XAResult (*Associate) (  
    XADTVSourceLocalItf self,  
    XADTVLocalAssociateInfo * pLocalAssociateInfo,  
    XAuint32 * pBearerID  
);
```

**Description**
Associates a local resource as defined in the `pLocalAssociateInfo` parameter. The soft bearers do not have static bearer IDs, so after a successful connection the `pBearerID` parameter value is generated and returned.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLocalAssociateInfo</td>
<td>[in]</td>
<td>Holds information needed to associate a local resource. The parameter identifies a URI.</td>
</tr>
<tr>
<td>pBearerID</td>
<td>[out]</td>
<td>Holds, after a successful association, the associated <code>bearerID</code> for the connected local resource.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_RESULT_CONTENT_NOT_FOUND`

**Comments**
-

#### Disassociate

```c
XAResult (*Disassociate) (  
    XADTVSourceLocalItf self,  
    XAuint32 bearerID  
);
```

**Description**
Disassociates a local resource with a specific local `bearerID`.

**Pre-conditions**
The bearer identified by the `bearerID` parameter must be associated to a local resource.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>Identifies a local resource that should be disassociated.</td>
</tr>
</tbody>
</table>
## Disassociate

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PRECONDITIONS_VIOLATED</td>
</tr>
</tbody>
</table>

| Comments | - |

## IsAssociated

```c
XAResult (*IsAssociated) (XADTVSourceLocalItf self, XAboolean * pIsAssociated);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Queries to find out if a local resource is associated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>None.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pIsAssociated</td>
<td>[out]</td>
<td>Indicates whether the local resource is associated or not.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| Comments | |

| See Also | |
7.12. XADTVSourceMulticastItf

Description
This interface represents bearers that rely on IP multicast delivery.
Examples of such bearers are regular IGMP Multicast IPTV or MBMS transmissions.
This interface holds methods specific to the multicast family.
This interface is a mandated interface on the DTVSource (See section 0) object. If one bearer technology is supported all other bearer technology interfaces (XADTVSourceBroadcastItf, XADTVSourceLocalItf, XADTVSourceMulticastItf and XADTVSourceUnicastItf) are optional to support.

Prototype
XA_API extern const XAInterfaceID XA_IID_DTVSOURCEMULTICAST;

struct XADTVSourceMulticastItf;
typedef const struct XADTVSourceMulticastItf * const * XADTVSourceMulticastItf;

struct XADTVSourceMulticastItf__ {  
    XAResult (*QuerySupportedMulticastTechnologies) (  
        XADTVSourceMulticastItf self,  
        XAuint32 * pNumberOfTechnologies,  
        XAuint32 * pTechnologies  
    );
    XAResult (*JoinMulticastGroup) (  
        XADTVSourceMulticastItf self,  
        XADTVMulticastGroupInfo * pMulticastGroupInfo,  
        XAuint32 * pBearerID  
    );
    XAResult (*LeaveMulticastGroup) (  
        XADTVSourceMulticastItf self,  
        XAuint32 bearerID  
    );
    XAResult (*GetMulticastBearerConnectionState) (  
        XADTVSourceMulticastItf self,  
        XAuint32 bearerID,  
        XAuint32 * pState  
    );
    XAResult (*RegisterMulticastConnectionStateChangedCallback) (  
        XADTVSourceMulticastItf self,  
        xaDTVMulticastConnectionStateChangedCallback Callback,  
        void * pContext  
    );
};

Interface ID
69558c98-cea5-48e7-88df-6bc1b12cde52
Callbacks

**xaDTVMulticastConnectionStateChangedCallback**

typedef void (XAAPIENTRY *
xaDTVMulticastConnectionStateChangedCallback) (
    XADTVSourceMulticastItf Caller,
    XAuint32 bearerID,
    XAuint32 connectionState,
    void * pContext
    );

<table>
<thead>
<tr>
<th>Description</th>
<th>Notifies the application that connection state has changed for a specific multicast bearer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Caller</td>
<td>[in] Interface on which this callback was registered.</td>
</tr>
<tr>
<td>bearerID</td>
<td>[in] Identifies the bearer for which the state changed.</td>
</tr>
<tr>
<td>connectionState</td>
<td>[in] The new connection state. See the XA_DTV_MULTICAST_CONNECTION_STATE macros for more information.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in] User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

Comments

See Also

Methods

**QuerySupportedMulticastTechnologies**

XAResult (*QuerySupportedMulticastTechnologies) ( XADTVSourceMulticastItf self,
    XAuint32 * pNumberOfTechnologies,
    XAuint32 * pTechnologies
    );

<table>
<thead>
<tr>
<th>Description</th>
<th>Queries the system for supported multicast technologies. The returned values represent various multicast technologies. This specification only represents MBMS and regular IGMP IPTV by default. The values supported could easily be extended and added to represent more technologies as they become available or defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>None.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>pNumberOfTechnologies</td>
<td>[in/out] As an input, specifies the length of the pTechnologies array. As an output, specifies the number of technologies currently available in the system and the length of the valid positions in the returned array. Returns 0 if no multicast technologies are available in the system.</td>
</tr>
</tbody>
</table>
### QuerySupportedMulticastTechnologies

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pTechnologies</td>
<td>Array of the multicast technologies currently available in the system. This parameter is populated by the call with the array of technologies, provided that the input value of pNumberOfTechnologies is equal to or greater than the number of actual technologies. If pNumberOfTechnologies is less than the number of actual input technologies, the error code XA_RESULT_BUFFER_INSUFFICIENT is returned. See the XA_DTV_MULTICAST_TECHNOLOGY macros for more information.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_BUFFER_INSUFFICIENT

**Comments**

The caller of this method is responsible for allocating the pTechnologies array.

### JoinMulticastGroup

```c
XAResult (*JoinMulticastGroup) (  
    XADTVSourceMulticastItf self,  
    XADTVMulticastGroupInfo * pMulticastGroupInfo,  
    XAuint32 * pBearerID  
);
```

**Description**

Joins a specific multicast group as defined in the pMulticastGroupInfo parameter. The soft bearers do not have static bearer IDs, so after a successful join the pBearerID parameter value is generated and returned.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pMulticastGroupInfo</td>
<td>[in] Holds information needed to join a specific multicast group. The parameter identifies a multicast technology and group information.</td>
</tr>
<tr>
<td>pBearerID</td>
<td>[out] Holds, after a successful join, the associated bearerID for the connected multicast transmission.</td>
</tr>
</tbody>
</table>
### JoinMulticastGroup

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_CONTENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

| Comments | -                                           |

### LeaveMulticastGroup

```c
XAResult (*LeaveMulticastGroup) ( 
   XADTVSourceMulticastItf self, 
   XAuint32 bearerID 
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Leaves a multicast group associated with a bearerID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>The bearer identified by the bearerID parameter must be a member of a multicast group.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bearerID</td>
<td>[in]</td>
<td>Identifies a multicast bearer that should leave its associated multicast group.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PRECONDITIONS_VIOLATED</td>
</tr>
</tbody>
</table>

| Comments | -                                           |

### GetMulticastBearerConnectionState

```c
XAResult (*GetMulticastBearerConnectionState) ( 
   XADTVSourceMulticastItf self, 
   XAuint32 bearerID, 
   XAuint32 * pState 
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Gets the connection state and status for a multicast bearer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>None.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bearerID</td>
<td>[in]</td>
<td>Identifies a multicast bearer as returned by the JoinMulticastGroup.</td>
</tr>
</tbody>
</table>

<p>| pState | [out] | The connection state as defined by the XA_DTV_MULTICAST_CONNECTION_STATE macros. |</p>
<table>
<thead>
<tr>
<th><strong>GetMulticastBearerConnectionState</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Return value</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Comments</strong></td>
</tr>
</tbody>
</table>
7.13. XADTVSourceUnicastItf

Description

This interface represents bearers that rely on IP unicast delivery.

Examples of such bearers are RTSP or HTTP transmissions. This interface holds methods specific to the unicast family.

This interface is a mandated interface on the DTVSource (See section 0) object. If one bearer technology is supported all other bearer technology interfaces (XADTVSourceBroadcastItf, XADTVSourceLocalItf, XADTVSourceMulticastItf and XADTVSourceUnicastItf) are optional to support.

Prototype

XA_API extern const XAInterfaceID XA_IID_DTVSOURCEUNICAST;

struct XADTVSourceUnicastItf_;
typedef const struct XADTVSourceUnicastItf * const * XADTVSourceUnicastItf;

struct XADTVSourceUnicastItf_ {
    XAResult (*QuerySupportedUnicastTechnologies) (XADTVSourceUnicastItf self, XAuint32 * pNumberOfTechnologies, XAuint32 * pTechnologies);
    XAResult (*Connect) (XADTVSourceUnicastItf self, XADTVUnicastConnectInfo * pUnicastConnectInfo, XAuint32 * pBearerID);
    XAResult (*Disconnect) (XADTVSourceUnicastItf self, XAuint32 bearerID);
    XAResult (*GetUnicastBearerConnectionState) (XADTVSourceUnicastItf self, XAuint32 bearerID, XAuint32 * pState);
    XAResult (*RegisterUnicastConnectionStateChangedCallback) (XADTVSourceUnicastItf self, xaDTVUnicastConnectionStateChangedCallback Callback, void * pContext);
};

Interface ID

e89cabe-9007-433b-9a57-4db4ade04ed3
## Callbacks

### xaDTVUnicastConnectionStateChangedCallback

```c
typedef void (XAPIENTRY * xaDTVUnicastConnectionStateChangedCallback)(
    XADTVSourceUnicastItf Caller,
    XAuint32 bearerID,
    XAuint32 connectionState,
    void * pContext
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Notifies the application that connection state has changed for a specific unicast bearer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>[in] Interface on which this callback was registered.</td>
</tr>
<tr>
<td>Caller</td>
<td>[in] Identifies the bearer for which the state changed.</td>
</tr>
<tr>
<td>bearerID</td>
<td>[in] The new connection state. See the XA_DTV_UNICAST_CONNECTION_STATE macros for more information.</td>
</tr>
<tr>
<td>connectionState</td>
<td>[in] User context data that is supplied when the callback method is registered.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in]</td>
</tr>
</tbody>
</table>

### See Also

### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Parameters</th>
<th>Parameters Description</th>
<th>Return value</th>
<th>Comments</th>
</tr>
</thead>
</table>
| **QuerySupportedUnicastTechnologies** | Queries the system for supported unicast technologies. The returned values represent various unicast technologies. This specification only represents RTSP and HTTP by default. The values supported could easily be extended and added to represent more technologies as they become available or defined. | self | [in] Interface self-reference. | The return value can be one of the following:  
- XA_RESULT_SUCCESS  
- XA_RESULT_PARAMETER_INVALID  
- XA_RESULT_BUFFER_INSUFFICIENT | The caller of this method is responsible for allocating the pTechnologies array. |
| | | pNumberOfTechnologies | [in/out] As an input, specifies the length of the pTechnologies array. As an output, specifies the number of technologies currently available in the system and the length of the valid positions in the returned array. Returns 0 if no unicast technologies are available in the system. If 0 is returned the input array is left untouched. | | |
| | | pTechnologies | [out] Array of the unicast technologies currently available in the system. This parameter is populated by the call with the array of technologies, provided that the input value of pNumberOfTechnologies is equal to or greater than the number of actual technologies. If pNumberOfTechnologies is less than the number of actual input technologies, the error code XA_RESULT_BUFFER_INSUFFICIENT is returned. See the XA_DTV_UNICAST_TECHNOLOGY macros for more information. | | |

**Pre-conditions** None.

**Parameters**

- **pNumberOfTechnologies** [in/out] As an input, specifies the length of the pTechnologies array. As an output, specifies the number of technologies currently available in the system and the length of the valid positions in the returned array. Returns 0 if no unicast technologies are available in the system. If 0 is returned the input array is left untouched.
- **pTechnologies** [out] Array of the unicast technologies currently available in the system. This parameter is populated by the call with the array of technologies, provided that the input value of pNumberOfTechnologies is equal to or greater than the number of actual technologies. If pNumberOfTechnologies is less than the number of actual input technologies, the error code XA_RESULT_BUFFER_INSUFFICIENT is returned. See the XA_DTV_UNICAST_TECHNOLOGY macros for more information.
### Connect

**XAResult (*Connect) (**

XADTVSourceUnicastItf self,
XADTVUnicastConnectInfo * pUnicastConnectInfo,
XAuint32 * pBearerID
**);

**Description**

Connects to a specific unicast server as defined in the pUnicastConnectInfo parameter. The soft bearers do not have static bearer IDs, so after a successful connection the pBearerID parameter value is generated and returned.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pUnicastConnectInfo</td>
<td>[in]</td>
<td>Holds information needed to connect to a specific unicast server. The parameter identifies a unicast technology and URI.</td>
</tr>
<tr>
<td>pBearerID</td>
<td>[out]</td>
<td>After a successful join holds the associated bearerID for the connected unicast transmission.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_CONTENT_NOT_FOUND

**Comments**

-

### Disconnect

**XAResult (*Disconnect) (**

XADTVSourceUnicastItf self,
XAuint32 bearerID
**);

**Description**

Disconnects from the unicast server associated with a bearerID.

**Pre-conditions**

The bearer identified by the bearerID parameter must be connected to a unicast server.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>Identifies a unicast bearer that should disconnect from the server.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID
- XA_RESULT_PRECONDITIONS_VIOLATED

**Comments**

-
### GetUnicastBearerConnectionState

```c
XAResult (*GetUnicastBearerConnectionState) (
    XADTVSourceUnicastItf self,
    XAuint32 bearerID,
    XAuint32 * pState
);
```

**Description**
Gets the connection state and status for a unicast bearer.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>Identifies a unicast bearer as returned by the Connect method.</td>
</tr>
<tr>
<td>pState</td>
<td>[out]</td>
<td>The connection state as defined by the XA_DTV_UNICAST_CONNECTION_STATE macros.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**

### RegisterUnicastConnectionStateChangedCallback

```c
XAResult (*RegisterUnicastConnectionStateChangedCallback) (
    XADTVSourceUnicastItf self,
    xaDTVUnicastConnectionStateChangedCallback Callback,
    void * pContext
);
```

**Description**
This method registers a callback event function that will receive callbacks related to changed connection state for a unicast bearer.

**Pre-conditions**
None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback</td>
<td>[in]</td>
<td>Identifies the callback function to receive callbacks about the unicast bearer connection state.</td>
</tr>
<tr>
<td>pContext</td>
<td>[in]</td>
<td>User context data that is supplied when the callback method is registered.</td>
</tr>
</tbody>
</table>

**Return value**
The return value can be one of the following:
- XA_RESULT_SUCCESS
- XA_RESULT_PARAMETER_INVALID

**Comments**
7.14. **XADTVSourceUtilitiesItf**

**Description**

*XADTVSourceUtilitiesItf* is used to query and control the bearers.

This interface is a mandated interface on the DTVSource (See section 0) object.
Prototype

XA_API extern const XAInterfaceID XA_IID_DTVSOURCEUTILITIES;

struct XADTVSourceUtilitiesItf_;
typedef const struct XADTVSourceUtilitiesItf_ * const *
XADTVSourceUtilitiesItf;

struct XADTVSourceUtilitiesItf_ {
    XAResult (*GetNumberOfAvailableStaticBearers) (  
        XADTVSourceUtilitiesItf self,  
        XAuint32 * pNumberOfStaticBearers,  
        XAuint32 * pBearerIDs
    );
    XAResult (*GetBearerInformation) (  
        XADTVSourceUtilitiesItf self,  
        XAuint32 bearerID,  
        XADTVSourceBearerInfo * pBearerInformation
    );
    XAResult (*SetBearerState) {  
        XADTVSourceUtilitiesItf self,  
        XAuint32 bearerID,  
        XAuint32 powerState
    };
    XAResult (*GetSignalStrength) {  
        XADTVSourceUtilitiesItf self,  
        XAuint32 bearerID,  
        XApermille SignalStrength
    };
    XAResult (*GetNetworkTime) {  
        XADTVSourceUtilitiesItf self,  
        XAuint32 bearerID,  
        XAtime * pTime
    };
    XAResult (*GetBearerState) {  
        XADTVSourceUtilitiesItf self,  
        XAuint32 bearerID,  
        XAuint32 powerState
    };
    XAResult (*QueryInternetConnections) {  
        XADTVSourceUtilitiesItf self,  
        XAuint32 * pNumberOfInternetConnections  
        XADTVInternetConnectionInfo * pInternetConnections
    };
    XAResult (*RegisterBearerChangeCallback) {  
        XADTVSourceUtilitiesItf self,  
        xADTVSourceBearerChangeCallback Callback,  
        void * pContext
    };
};

Interface ID

362515a0-c463-11de-8a39-0800200c9a66
## xaDTVSourceBearerChangeCallback

```c
typedef void (XAENTRY * xaDTVSourceBearerChangeCallback)(
    XADTVSourceUtilitiesItf Caller,
    XAuint32 bearerID,
    XAuint32 changeType,
    void * pContext
);
```

### Description

Notifies the application that a bearer has been added or removed, or that the bearer state has changed.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller</td>
<td>[in] Interface on which this callback was registered.</td>
<td></td>
</tr>
<tr>
<td>bearerID</td>
<td>[in] Identifies the bearer for which the state changed.</td>
<td></td>
</tr>
<tr>
<td>changeType</td>
<td>[in] Identifies the type of change that occurred on the bearer. See the XA_DTV_SOURCE_BEARER_CHANGE macro for more information.</td>
<td></td>
</tr>
<tr>
<td>pContext</td>
<td>[in] User context data that is supplied when the callback method is registered.</td>
<td></td>
</tr>
</tbody>
</table>

### Comments

None

### See Also
**Methods**

### GetNumberAvailableStaticBearers

```c
XAresult (*GetNumberAvailableStaticBearers) (  
  XADTVSourceUtilitiesItf self,  
  XAuInt32 * pNumberOfStaticBearers,  
  XAuInt32 * pBearerIDs  
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Queries information about the number of statically available bearers.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The statically available bearers in the system are returned as an array of bearer IDs.</td>
</tr>
<tr>
<td></td>
<td>Each available bearer has its own ID, meaning that if there exist more than one bearer of the same type they will still have their own, unique, ID.</td>
</tr>
<tr>
<td></td>
<td>Soft bearers, such as RTSP/Unicast bearers, are not identified in this list. Soft bearers are dynamically created and returned when connected to multicast groups, unicast servers or local resources.</td>
</tr>
<tr>
<td></td>
<td>It is up to the application to allocate the pBearerIDs array and the pNumberOfBearers parameter prior to calling the method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pre-conditions</th>
<th>None.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pNumberOfBearers</td>
<td>[in/out] As an input, specifies the length of the pBearerIDs array. As an output, specifies the number of bearers currently available in the system and the length of the valid positions in the returned array. Returns 0 if no bearers are available in the system.</td>
</tr>
<tr>
<td>pBearerIDs</td>
<td>[out] Array of the Bearer IDs currently available in the system. This parameter is populated by the call with the array of bearer device IDs, provided that the input value of pNumberOfBearers is equal to or greater than the number of actual input bearer IDs. If pNumberOfBearers is less than the number of actual input device IDs, the error code XA_RESULT_BUFFER_INSUFFICIENT is returned.</td>
</tr>
</tbody>
</table>
### GetNumberAvailableStaticBearers

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_BUFFER_INSUFFICIENT</td>
</tr>
</tbody>
</table>

| Comments | The bearer ID of the bearers is static, unique and will not change during the application lifetime. The caller of this method is responsible for allocating the pBearerIDs array. |

### GetBearerInformation

```c
XAresult (*GetBearerInformation) ( XADTVSourceUtilitiesItf self, XAuint32 bearerID, XADTVSourceBearerInfo * pBearerInformation );
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Queries information about the specified bearer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method takes a bearer ID as input and returns information about that bearer.</td>
</tr>
<tr>
<td></td>
<td>Available static bearer IDs in the system may be queried using the GetNumberAvailableStaticBearers method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preconditions</th>
<th>None.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bearerID</td>
<td>[in]</td>
<td>The ID of the bearer.</td>
</tr>
<tr>
<td></td>
<td>pBearerInformation</td>
<td>[out]</td>
<td>Structure containing the bearer information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return value</th>
<th>The return value can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
</tbody>
</table>

| Comments | None. |
### SetBearerState

**XResult (*SetBearerState) {**

- XADTVSourceUtilitiesItf self,
- XUint32 bearerID,
- XUint32 powerState

**};**

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets the specified bearer to the specified power state.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>None.</td>
</tr>
<tr>
<td></td>
<td>powerState [in] Designator for the power state of the bearer. See the XA_DTV_BEARER_STATE macros for details.</td>
</tr>
<tr>
<td>Return value</td>
<td>The return value can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>- XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>- XA_RESULT_UNSUPPORTED</td>
</tr>
<tr>
<td>Comments</td>
<td>XA_RESULT_UNSUPPORTED is returned if the IDLE state is not supported when trying to set it.</td>
</tr>
<tr>
<td>See Also</td>
<td>GetBearerState</td>
</tr>
</tbody>
</table>

### GetSignalStrength

**XResult (*GetSignalStrength) {**

- XADTVSourceUtilitiesItf self,
- XUint32 bearerID,
- XApermille * SignalStrength

**};**

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the current signal strength of a tuned bearer. For local bearers the signal strength is always 100%.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>None.</td>
</tr>
<tr>
<td></td>
<td>pSignalStrength [out] The indicated signal strength.</td>
</tr>
<tr>
<td>Return value</td>
<td>The return value can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>- XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td></td>
<td>- XA_RESULT_PRECONDITIONS_VIOLATED – Hardware not enabled or not tuned</td>
</tr>
<tr>
<td>Comments</td>
<td>A tuned bearer is a bearer that has been tuned using a ScanInfo to actively set the tuner to a specific frequency or resource.</td>
</tr>
</tbody>
</table>
### GetNetworkTime

**XAResult (*GetNetworkTime) (**

- `XADTVSourceUtilitiesItf self`
- `XAuint32 bearerID`
- `XAtime * pTime`

**;**

**Description**

Returns the network time for the specified bearer.

If the specified bearer does not have any kind of network time reference, the current system time is returned instead.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerID</td>
<td>[in]</td>
<td>The ID of the bearer.</td>
</tr>
<tr>
<td>pTime</td>
<td>[out]</td>
<td>The network time associated with the specified bearer.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_RESULT_PRECONDITIONS_VIOLATED` - Hardware not enabled or not tuned.

**Comments**

None

### GetBearerState

**XAResult (*GetBearerState) (**

- `XADTVSourceUtilitiesItf self`
- `XAuint32 bearerID`
- `XAuint32 powerState`

**;**

**Description**

Gets the power state of the specified bearer.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>powerState</td>
<td>[out]</td>
<td>Designator for the power state of the bearer.</td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
### QueryInternetConnections

```c
XAResult (*QueryInternetConnections) ( 
    XADTVSourceUtilitiesItf self, 
    XAuint32 * pNumberOfInternetConnections 
    XADTVInternetConnectionInfo * pInternetConnections 
);
```

**Description**

Queries for all available internet connections.

The internet connections returned by this method are only used when connecting a multicast or unicast connection using the `XADTVSourceMulticastItf::Join` or `XADTVSourceUnicastItf::Connect` methods. The structures which use the internet connection are `XADTVMulticastGroupInfo` and `XADTVUnicastConnectInfo`.

**Pre-conditions**

None.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pNumberOfInternetConnections</td>
<td>[in/out] As an in-parameter this parameter defines the length of the preallocated pInternetConnections array. As an out-parameter this parameter is set to the number of available internet connections in the system.</td>
<td></td>
</tr>
<tr>
<td>pInternetConnections</td>
<td>[out] An array of all internet connections currently available in the system. If this parameter is set to NULL, the pNumberOfInternetConnections parameter will be set to the number of available connections. The number of internet connections available is variable and might change during run-time. This implies that the pNumberOfInternetConnections parameter might be invalid and should only be used as an indication of the amount of connections available.</td>
<td></td>
</tr>
</tbody>
</table>

**Return value**

The return value can be one of the following:

- `XA_RESULT_SUCCESS`
- `XA_RESULT_PARAMETER_INVALID`
- `XA_BUFFER_INSUFFICIENT`

**Comments**

The caller of this method is responsible for allocating the pInternetConnections array.

**See Also**

- `XADTVSourceMulticastItf::Join`
- `XADTVSourceUnicastItf::Connect`
- `XADTVMulticastGroupInfo`
- `XADTVUnicastConnectInfo`
### RegisterBearerChangeCallback

```c
XIAresult (*RegisterBearerChangeCallback) ( 
    XADTVSourceUtilitiesItf self, 
    xaDTVSourcedeviceBearerChangeCallback Callback, 
    void * pContext 
);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets the callback for bearer property change event notifications. Is generated when a bearer is added, removed or when bearer state changes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-conditions</td>
<td>None</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Callback</strong></td>
<td>[in] Specifies the callback method.</td>
</tr>
<tr>
<td><strong>pContext</strong></td>
<td>[in] User context data that is supplied when the callback method is registered.</td>
</tr>
<tr>
<td><strong>Return value</strong></td>
<td>The return value can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_SUCCESS</td>
</tr>
<tr>
<td></td>
<td>• XA_RESULT_PARAMETER_INVALID</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>See Also</strong></td>
<td>xaDTVSourcedeviceBearerChangeCallback()</td>
</tr>
</tbody>
</table>
8. Macros and Typedefs

8.1. Structures

8.1.1. XADataLocator_DTVService

typedef struct XADataLocator_DTVService_ {
  XAuint32 locatorType,
  XAObjectItf service
} XADataLocator_DTVService;

Fields include:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>locatorType</td>
<td>Locator type, which must be XA_DATALOCATOR_DTVSERVICE for this structure.</td>
</tr>
<tr>
<td>service</td>
<td>The DTV service object created by the Engine.</td>
</tr>
</tbody>
</table>

8.1.2. XADTVContentProtectionInfo

typedef struct XADTVContentProtectionInfo_ {
   XAchar * pUri,
   XAuuint32 reason,
   XAuuint32 streamIndex
} XADTVContentProtectionInfo;

This structure is used for reporting a decryption error due to missing rights.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pUri</td>
<td>The URI needed to get new or updated rights for the streams</td>
</tr>
<tr>
<td>reason</td>
<td>The reason for the event. Refer to the XA_DTV_CONTENT_PROTECTION macros.</td>
</tr>
<tr>
<td>streamIndex</td>
<td>This field indicates the stream index for the stream missing rights. To find out what kind of stream and to retrieve more information about the stream, use the XAStreamInformationItf together with this index parameter.</td>
</tr>
</tbody>
</table>

8.1.3. XADTVInternetConnectionInfo

typedef struct XADTVInternetConnectionInfo_ {
   XAchar connectionName[256],
   void * pInternetConnection
} XADTVInternetConnectionInfo;

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectionName</td>
<td>The NULL-terminated plaintext name of this internet connection.</td>
</tr>
</tbody>
</table>
### 8.1.4. XADTVLocalAssociateInfo

typedef struct XADTVLocalAssociateInfo_ {
    XAchar uri[256];
} XADTVLocalAssociateInfo;

XADTVLocalAssociateInfo represents a local association URI.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>The URI to the local resource. This could point to a file in the file system.</td>
</tr>
</tbody>
</table>

### 8.1.5. XADTVMulticastGroupInfo

typedef struct XADTVMulticastGroupInfo_ {
    XAuint32 technologyIdentifier;
    void * pInternetConnection;
    XAchar multicastGroupURL[256];
} XADTVMulticastGroupInfo;

XADTVMulticastGroupInfo represents a multicast group to join.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>technologyIdentifier</td>
<td>Identifies a multicast technology as defined by the XA_DTV_MULTICAST_TECHNOLOGY macros.</td>
</tr>
<tr>
<td>pInternetConnection</td>
<td>The system specific internet connection pointer. See the XADTVSourceUtilitiesItf::QueryInternetConnections method for more information on retrieving system internet connections.</td>
</tr>
<tr>
<td>multicastGroupURL</td>
<td>Information needed to join a specific multicast group. (example: udp://224.0.1.1:50000)</td>
</tr>
</tbody>
</table>

The multicastGroupURL member holds the multicast connection information needed to join a specific multicast group.

### 8.1.6. XADTVServiceConnectionInfo

typedef struct XADTVServiceConnectionInfo_ {
    XAchar serviceName[256],
    XAuint32 IDType,
    XAuint32 ID,
    void * pConnectionData
} XADTVServiceConnectionInfo;

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name identifier of the service.</td>
</tr>
</tbody>
</table>
### Field

<table>
<thead>
<tr>
<th>IDType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicates the type of ID supplied. See the XA_DTV_EPG_CONTENT_ID_TYPE macros for more information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>The ID for the content. This ID cannot be a bundle ID.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pConnectionData</th>
</tr>
</thead>
<tbody>
<tr>
<td>The implementation-specific connection data. This data is only used as an input for the service to pass on to the DTVSource object. The data in this parameter might vary depending on bearer type used. This data is not meant to be parsed by the application.</td>
</tr>
</tbody>
</table>

#### 8.1.7. XADTVServiceDataDeliveryFileDescriptor

```c
typedef struct XADTVServiceDataDeliveryFileDescriptor_ {
    XAchar * pFileName,
    XAuuint32 Version,
    XAuuint32 FileLength,
    XAchar * pMIMEType
} XADTVServiceDataDeliveryFileDescriptor;
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pFileName</td>
<td>The filename of the file.</td>
</tr>
<tr>
<td>Version</td>
<td>The version number of the file.</td>
</tr>
<tr>
<td>FileLength</td>
<td>The length of the file in bytes.</td>
</tr>
<tr>
<td>pMIMEType</td>
<td>The associated MIME-type of the file.</td>
</tr>
</tbody>
</table>

#### 8.1.8. XADTVServiceTimeShiftInfo

```c
typedef struct XADTVServiceTimeShiftInfo_ {
    XAtime DataStartTime;
    XAtime DataEndTime;
    XAuuint32 BufferBytesUsed;
    XAuuint32 BufferBytesAvailable;
    XAuuint32 BufferSecondsUsed;
    XAuuint32 BufferSecondsAvailable;
} XADTVServiceTimeShiftInfo;
```

This structure is used for reporting time shift information.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataStartTime</td>
<td>The system timestamp for the start of the data in the time shift buffer</td>
</tr>
<tr>
<td>DataEndTime</td>
<td>The system timestamp for the end of the data in the time shift buffer</td>
</tr>
<tr>
<td>BufferBytesUsed</td>
<td>How many bytes that have been utilized of the time shift buffer up until now</td>
</tr>
<tr>
<td>BufferBytesAvailable</td>
<td>How many bytes that are available in the time shift buffer</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>up until now</td>
<td>The amount data, in seconds, currently used in the time shift buffer</td>
</tr>
<tr>
<td>BufferSecondsUsed</td>
<td>An estimate of the amount of seconds more that will fit in the time shift buffer</td>
</tr>
</tbody>
</table>

### 8.1.9. XADTVSourceBearerInfo

```c
typedef struct XADTVSourceBearerInfo_ {
    XAuint32 bearerId,
    XAchar bearerName[256],
    XAuint32 bearerTechnology,
    XAuint32 bearerFamily,
    XAuint32 bearerType,
    XAuint32 bearerVersion,
    XAboolean bearerLocked
} XADTVSourceBearerInfo;
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerId</td>
<td>The bearer ID associated with this bearer information.</td>
</tr>
<tr>
<td>bearerName</td>
<td>The NULL-terminated name of this bearer.</td>
</tr>
<tr>
<td>bearerTechnology</td>
<td>The bearer technology associated with this bearer. See the</td>
</tr>
<tr>
<td></td>
<td>XA_DTV_BEARER_TECHNOLOGY macros for more information</td>
</tr>
<tr>
<td>bearerFamily</td>
<td>The bearer family associated with this bearer. See the</td>
</tr>
<tr>
<td></td>
<td>XA_DTV_BEARER_FAMILY macros for more information</td>
</tr>
<tr>
<td>bearerType</td>
<td>The bearer type associated with this bearer. See the</td>
</tr>
<tr>
<td></td>
<td>XA_DTV_BEARER_TYPE macros for more information</td>
</tr>
<tr>
<td>bearerVersion</td>
<td>The version of the bearer. This indicates, for example, that the</td>
</tr>
<tr>
<td></td>
<td>bearer is a DVB-T2 bearer. The technology, family and type are</td>
</tr>
<tr>
<td></td>
<td>the same as for DVB-T.</td>
</tr>
<tr>
<td>bearerLocked</td>
<td>Indicates whether the specified bearer is locked to a specific</td>
</tr>
<tr>
<td></td>
<td>service-provider or platform. Such bearers might have</td>
</tr>
<tr>
<td></td>
<td>limitations associated with the tuning.</td>
</tr>
</tbody>
</table>

### 8.1.10. XADTVSourceScanInfo

```c
typedef struct XADTVSourceScanInfo_ {
    XAuint32 bearerId,
    XAchar * pScanName,
```
typedef struct XADTVSourceScanInfo {
    XAuint32 scanNameLength,
    XAboolean singleService,
    Xachar * pMultiplexMIMEType,
    Xauint32 multiplexMIMETypeLength,
    void * pBearerSpecificScanData
} XADTVSourceScanInfo;

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bearerId</td>
<td>The bearer ID associated with this scan info</td>
</tr>
<tr>
<td>pScanName</td>
<td>The plaintext name of this scan info. Could be set to channel name, platform provider name or any other description of this specific scan info</td>
</tr>
<tr>
<td>scanNameLength</td>
<td>The length of the pScanName parameter</td>
</tr>
<tr>
<td>singleService</td>
<td>This member parameter indicates if the scan info represents a single service or a complete set of multiple services, for instance a DVB-H platform</td>
</tr>
<tr>
<td>pMultiplexMIMEType</td>
<td>The MIME-type, identifying the multiplex type. This field could be empty if MIME-type could not be identified during scan.</td>
</tr>
<tr>
<td>multiplexMIMETypeLengt</td>
<td>h</td>
</tr>
<tr>
<td>pBearerSpecificScanData</td>
<td>A pointer to the bearer specific data needed for the specified bearer to tune to this service at a later point in time. The data in the returned struct can be used in the SetScanInfo method to select the service.</td>
</tr>
<tr>
<td></td>
<td>Depending on the type of bearer, this struct can contain information such as platform ID (for DVB-H), a plain frequency (for ISDB-T) or any other bearer specific tune data. For bearers containing a large multiplex with many services on one frequency, this data must not only specify the frequency but also the PID of the specific service within the multiplex.</td>
</tr>
<tr>
<td></td>
<td>If the bearer specified by bearerId contains multiple services grouped together which cannot be identified at the time of the scan, the corresponding EPG is required to select a particular service. In this case, only one callback will be returned per identified group of services.</td>
</tr>
<tr>
<td></td>
<td>When a group of services are returned in a single callback, the singleService parameter will be set to FALSE.</td>
</tr>
</tbody>
</table>

8.1.11. XADTVUnicastConnectInfo

typedef struct XADTVUnicastConnectInfo_ {
    XAuint32 technologyIdentifier;
}
void * pInternetConnection;
XAchar unicastURL[256];
} XADTVUnicastConnectInfo;

XADTVUnicastConnectInfo represents a unicast server and technology.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>technologyIdentifier</td>
<td>Identifies a unicast technology as defined by the XA_DTV_UNICAST_TECHNOLOGY macros.</td>
</tr>
<tr>
<td>pInternetConnection</td>
<td>The system specific internet connection pointer. See the XADTVSourceUtilitiesItf::QueryInternetConnections method for more information on retrieving system internet connections.</td>
</tr>
<tr>
<td>unicastURL</td>
<td>The URL to the unicast server. (example: rtsp://192.0.0.100:8000/example.3gp)</td>
</tr>
</tbody>
</table>
8.2. Macros

### 8.2.1. XA_DTV_EPG_ATTR_INT

```c
#define XA_DTV_EPG_ATTR_INT_PERMISSION ((XAuint32) 0x00000006)
#define XA_DTV_EPG_ATTR_INT_SERVICE_TYPE ((XAuint32) 0x00000020)
```

The macros are defined for integer attribute operations.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_EPG_ATTR_INT_PERMISSION</td>
<td>Use XA_DTV_EPG_PERMISSION macros as values.</td>
</tr>
<tr>
<td>XA_DTV_EPG_ATTR_INT_SERVICE_TYPE</td>
<td>Type of service.</td>
</tr>
</tbody>
</table>

### 8.2.2. XA_DTV_EPG_ATTR_INT_PRICE

```c
#define XA_DTV_EPG_ATTR_INT_PRICE_DECIMAL_PART ((XAuint32) 0x00000051)
#define XA_DTV_EPG_ATTR_INT_PRICE_INTEGER_PART ((XAuint32) 0x00000050)
```

The macros are defined for additional integer attributes for purchase bundles.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_EPG_ATTR_INT_PRICE_DECIMAL_PART</td>
<td>Decimal part of the price.</td>
</tr>
<tr>
<td>XA_DTV_EPG_ATTR_INT_PRICE_INTEGER_PART</td>
<td>Integer part of the price.</td>
</tr>
</tbody>
</table>

### 8.2.3. XA_DTV_EPG_ATTR_STR

```c
#define XA_DTV_EPG_ATTR_STR_DESCRIPTION ((XAuint32) 0x00000002)
#define XA_DTV_EPG_ATTR_STR_GENRE ((XAuint32) 0x00000003)
#define XA_DTV_EPG_ATTR_STR_NAME ((XAuint32) 0x00000001)
#define XA_DTV_EPG_ATTR_STR_PARENTAL_RATING ((XAuint32) 0x00000004)
#define XA_DTV_EPG_ATTR_STR_PARENTAL_RATING_SYSTEM ((XAuint32) 0x00000005)
#define XA_DTV_EPG_ATTR_STR_PRICE_CURRENCY ((XAuint32) 0x00000041)
#define XA_DTV_EPG_ATTR_STR_WEBSHOP_URL ((XAuint32) 0x00000040)
```

The macros are defined for string attributes.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_EPG_ATTR_STR_DESCRIPTION</td>
<td>Content description.</td>
</tr>
<tr>
<td>XA_DTV_EPG_ATTR_STR_GENRE</td>
<td>Content genre.</td>
</tr>
<tr>
<td>XA_DTV_EPG_ATTR_STR_NAME</td>
<td>Name of content.</td>
</tr>
<tr>
<td>XA_DTV_EPG_ATTR_STR_PARENTAL_RATING</td>
<td>The parental rating.</td>
</tr>
</tbody>
</table>
### 8.2.4. XA_DTV_EPG_ATTR_TIME

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_EPG_ATTR_TIME_END</td>
<td>End time.</td>
</tr>
<tr>
<td>XA_DTV_EPG_ATTR_TIME_START</td>
<td>Start time.</td>
</tr>
</tbody>
</table>

The macros are defined for time attributes (only for contents and purchase bundles).

### 8.2.5. XA_DTV_EPG_OPR_COMP

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_EPG_OPR_COMP_LESS_THAN</td>
<td>The first is less than the second.</td>
</tr>
<tr>
<td>XA_DTV_EPG_OPR_COMP_EQUAL</td>
<td>Both are equal.</td>
</tr>
<tr>
<td>XA_DTV_EPG_OPR_COMP_GREATER_THAN</td>
<td>The first is greater than the second.</td>
</tr>
</tbody>
</table>

The macros are defined for comparison operations.

### 8.2.6. XA_DTV_EPG_PERMISSION

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_EPG_PERMISSION_NOT_PROTECTED</td>
<td>Not protected (not subscribed).</td>
</tr>
<tr>
<td>XA_DTV_EPG_PERMISSION_NOT_SUBSCRIBED</td>
<td>Protected, currently not subscribed.</td>
</tr>
<tr>
<td>XA_DTV_EPG_PERMISSION_SUBSCRIBED</td>
<td>Protected, subscribed.</td>
</tr>
</tbody>
</table>

The macros are defined for permissions.
8.2.7. **XA_DTV_BEARER_FAMILY**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_BEARER_FAMILY_DVB</td>
<td>The DVB family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_ATSC</td>
<td>The ATSC family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_ISDB</td>
<td>The ISDB family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_IMB</td>
<td>The IMB family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_DMB</td>
<td>The DMB family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_CMMB</td>
<td>The CMMB family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_MBMS</td>
<td>The MBMS family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_RTSP</td>
<td>The RTSP family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_HTTP</td>
<td>The HTTP family.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_FAMILY_FILE</td>
<td>The file family.</td>
</tr>
</tbody>
</table>

The "soft" bearers RTSP, HTTP, FILE and the multicast bearers are higher level protocols. They exist as dynamically allocated bearers.

8.2.8. **XA_DTV_BEARER_STATE**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_BEARER_STATE_ON</td>
<td>The bearer is enabled</td>
</tr>
<tr>
<td>XA_DTV_BEARER_STATE_OFF</td>
<td>The bearer is disabled</td>
</tr>
<tr>
<td>XA_DTV_BEARER_STATE_IDLE</td>
<td>Standby mode</td>
</tr>
</tbody>
</table>

8.2.9. **XA_DTV_BEARER_TECHNOLOGY**

#define XA_DTV_BEARER_TECHNOLOGY_BROADCAST ((XAuint32) 0x00000000)
#define XA_DTV_BEARER_TECHNOLOGY_UNICAST ((XAuint32) 0x00000001)
#define XA_DTV_BEARER_TECHNOLOGY_MULTICAST ((XAuint32) 0x00000002)
#define XA_DTV_BEARER_TECHNOLOGY_LOCAL ((XAuint32) 0x00000003)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_BEARER_TECHNOLOGY_BROADCAST</td>
<td>Broadcast oriented bearer.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TECHNOLOGY_UNICAST</td>
<td>Unicast oriented bearer.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TECHNOLOGY_MULTICAST</td>
<td>Multicast oriented bearer.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TECHNOLOGY_LOCAL</td>
<td>Local delivery within the device itself. This could be delivery by a local file, for example a recording.</td>
</tr>
</tbody>
</table>

8.2.10. XA_DTV_BEARER_TYPE

#define XA_DTV_BEARER_TYPE_TERRESTRIAL        ((XAuint32) 0x00000000)
#define XA_DTV_BEARER_TYPE_CABLE               ((XAuint32) 0x00000001)
#define XA_DTV_BEARER_TYPE_HANDHELD            ((XAuint32) 0x00000002)
#define XA_DTV_BEARER_TYPE_SATELLITE           ((XAuint32) 0x00000003)
#define XA_DTV_BEARER_TYPE_SATELLITE_HANDHELD  ((XAuint32) 0x00000004)
#define XA_DTV_BEARER_TYPE_INTERNET            ((XAuint32) 0x00000005)
#define XA_DTV_BEARER_TYPE_FILE                ((XAuint32) 0x00000006)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_BEARER_TYPE_TERRESTRIAL</td>
<td>Terrestrial bearer type.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TYPE_CABLE</td>
<td>Cable based bearer type.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TYPE_HANDHELD</td>
<td>Handheld oriented bearer type.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TYPE_SATELLITE</td>
<td>Satellite bearer type.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TYPE_SATELLITE_HANDHELD</td>
<td>Satellite handheld bearer type.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TYPE_INTERNET</td>
<td>Internet bearer type.</td>
</tr>
<tr>
<td>XA_DTV_BEARER_TYPE_FILE</td>
<td>File bearer type.</td>
</tr>
</tbody>
</table>

8.2.11. XA_DTV_CONTENT_PROTECTION

#define XA_DTV_CONTENT_PROTECTION_MISSING_RIGHTS ((XAuint32) 0x00000001)
#define XA_DTV_CONTENT_PROTECTION_EXPIRED_RIGHTS ((XAuint32) 0x00000002)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_CONTENT_PROTECTION_MISSING_RIGHTS</td>
<td>Rights do not exist on the device to be able to decrypt the content</td>
</tr>
</tbody>
</table>
### 8.2.12. XA_DTV_EPG_CONTENT_ID_TYPE

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_EPG_CONTENT_ID_TYPE_BUNDLE</td>
<td>The ID specified refers to a bundle.</td>
</tr>
<tr>
<td>XA_DTV_EPG_CONTENT_ID_TYPE_SERVICE</td>
<td>The ID specified refers to a service.</td>
</tr>
<tr>
<td>XA_DTV_EPG_CONTENT_ID_TYPE_CONTENT</td>
<td>The ID specified refers to a specific content.</td>
</tr>
</tbody>
</table>

```c
#define XA_DTV_EPG_CONTENT_ID_TYPE_BUNDLE ((XAuint32) 0x00000002)
#define XA_DTV_EPG_CONTENT_ID_TYPE_SERVICE ((XAuint32) 0x00000001)
#define XA_DTV_EPG_CONTENT_ID_TYPE_CONTENT ((XAuint32) 0x00000000)
```

### 8.2.13. XA_DTV_MULTICAST_CONNECTION_STATE

The macros are defined to identify connection state of a multicast bearer.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_MULTICAST_CONNECTION_STATE_CONNECTED</td>
<td>Identifies the connected state for the multicast bearer. The bearer does not receive any data.</td>
</tr>
<tr>
<td>XA_DTV_MULTICAST_CONNECTION_STATE_CONNECTED_RECEIVING</td>
<td>Identifies the connected state for the multicast bearer. The bearer receives data.</td>
</tr>
<tr>
<td>XA_DTV_MULTICAST_CONNECTION_STATE_DISCONNECTED</td>
<td>Indicates that the bearer is disconnected.</td>
</tr>
</tbody>
</table>

```c
#define XA_DTV_MULTICAST_CONNECTION_STATE_CONNECTED ((XAuint32) 0x00000001)
#define XA_DTV_MULTICAST_CONNECTION_STATE_CONNECTED_RECEIVING ((XAuint32) 0x00000002)
#define XA_DTV_MULTICAST_CONNECTION_STATE_DISCONNECTED ((XAuint32) 0x00000003)
```

### 8.2.14. XA_DTV_MULTICAST_TECHNOLOGY

```c
#define XA_DTV_MULTICAST_TECHNOLOGY_IPTV ((XAuint32) 0x00000001)
```
The macros are defined to identify supported multicast technologies.

### 8.2.15. **XA_DTV_PLAYER_TIMED_OBJECTS**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_PLAYER_TIMED_OBJECTS_TEXT</td>
<td>The device is capable of rendering Text-based timed objects.</td>
</tr>
<tr>
<td>XA_DTV_PLAYER_TIMED_OBJECTS_GRAPHICS</td>
<td>The device is capable of rendering graphical timed objects.</td>
</tr>
<tr>
<td>XA_DTV_PLAYER_TIMED_OBJECTS_HTML</td>
<td>The device is capable of rendering HTML based timed objects.</td>
</tr>
<tr>
<td>XA_DTV_PLAYER_TIMED_OBJECTS_ALL</td>
<td>Let the device render all the timed object types it is capable of rendering.</td>
</tr>
<tr>
<td>XA_DTV_PLAYER_TIMED_OBJECTS_NONE</td>
<td>Don't render any timed objects or no timed objects rendering available.</td>
</tr>
</tbody>
</table>

### 8.2.16. **XA_DTV_RECORDER_MODE**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_RECORDER_MODE_ALL_STREAMS</td>
<td>No alterations of streams are performed. The file delivery carousel, all audio and video tracks, and all the timed text is saved.</td>
</tr>
<tr>
<td>XA_DTV_RECORDER_MODE_ACTIVE_STREAMS</td>
<td>Save only the selected streams to disk.</td>
</tr>
</tbody>
</table>

### 8.2.17. **XA_DTV_SERVICE_FILE_DL_PROGRESS**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
#define XA_DTV_SERVICE_FILE_DL_PROGRESS_ABORTED ((XAuint32) 0x00000003)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_SERVICE_FILE_DL_PROGRESS_DOWNLOADING</td>
<td>The file is in progress of being delivered.</td>
</tr>
<tr>
<td>XA_DTV_SERVICE_FILE_DL_PROGRESS_FINISHED</td>
<td>The delivery of the file has finished.</td>
</tr>
<tr>
<td>XA_DTV_SERVICE_FILE_DL_PROGRESS_ABORTED</td>
<td>The delivery of the file was aborted.</td>
</tr>
</tbody>
</table>

### 8.2.18. XA_DTV_SERVICE_FILE_DESCRIPTION

#define XA_DTV_SERVICE_FILE_DESCRIPTION_NEW ((XAuint32) 0x00000001)
#define XA_DTV_SERVICE_FILE_DESCRIPTION_UPDATED ((XAuint32) 0x00000002)
#define XA_DTV_SERVICE_FILE_DESCRIPTION_REMOVED ((XAuint32) 0x00000003)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_SERVICE_FILE_DESCRIPTION_NEW</td>
<td>A new file descriptor arrived in the stream.</td>
</tr>
<tr>
<td>XA_DTV_SERVICE_FILE_DESCRIPTION_UPDATED</td>
<td>A current file descriptor was updated.</td>
</tr>
<tr>
<td>XA_DTV_SERVICE_FILE_DESCRIPTION_REMOVED</td>
<td>A previous file descriptor was removed.</td>
</tr>
</tbody>
</table>

### 8.2.19. XA_DTV_SERVICE_TYPE

#define XA_DTV_SERVICE_TYPE_BASIC_RADIO ((XAuint32) 0x00000002)
#define XA_DTV_SERVICE_TYPE_BASIC_TV ((XAuint32) 0x00000001)
#define XA_DTV_SERVICE_TYPE_CACHECAST ((XAuint32) 0x00000004)
#define XA_DTV_SERVICE_TYPE_FILE_DOWNLOAD ((XAuint32) 0x00000005)
#define XA_DTV_SERVICE_TYPE_RESERVED1 ((XAuint32) 0x00000003)
#define XA_DTV_SERVICE_TYPE_UNSPECIFIED ((XAuint32) 0x00000000)

The macros are defined for service type.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_SERVICE_TYPE_BASIC_RADIO</td>
<td>Basic radio.</td>
</tr>
<tr>
<td>XA_DTV_SERVICE_TYPE_BASIC_TV</td>
<td>Basic TV.</td>
</tr>
<tr>
<td>XA_DTV_SERVICE_TYPE_PODCAST</td>
<td>A way to broadcast episodic programs. These podcasts are usually downloaded from a web page, but could also reside multiplexed in broadcasted data streams and downloaded using file delivery technology.</td>
</tr>
</tbody>
</table>
### 8.2.20. `XA_DTV_SOURCE_BEARER_CHANGE`

- `#define XA_DTV_SOURCE_BEARER_CHANGE_ADDED ((XAuint32) 0x00000001)`
- `#define XA_DTV_SOURCE_BEARER_CHANGE_REMOVED ((XAuint32) 0x00000002)`
- `#define XA_DTV_SOURCE_BEARER_CHANGE_STATE ((XAuint32) 0x00000003)`

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>XA_DTV_SOURCE_BEARER_CHANGE_ADDED</code></td>
<td>This macro indicates that a bearer has been added.</td>
</tr>
<tr>
<td><code>XA_DTV_SOURCE_BEARER_CHANGE_REMOVED</code></td>
<td>This macro indicates that a bearer has been removed.</td>
</tr>
<tr>
<td><code>XA_DTV_SOURCE_BEARER_CHANGE_STATE</code></td>
<td>This macro indicates that a bearer’s state has changed.</td>
</tr>
</tbody>
</table>

### 8.2.21. `XA_DTV_UNICAST_CONNECTION_STATE`

- `#define XA_DTV_UNICAST_CONNECTION_STATE_CONNECTED ((XAuint32) 0x00000001)`
- `#define XA_DTV_UNICAST_CONNECTION_STATE_CONNECTED_RECEIVING ((XAuint32) 0x00000002)`
- `#define XA_DTV_UNICAST_CONNECTION_STATE_DISCONNECTED ((XAuint32) 0x00000003)`

The macros are defined to identify the connection state.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>XA_DTV_UNICAST_CONNECTION_STATE_CONNECTED</code></td>
<td>Identifies the connected state for a unicast bearer.</td>
</tr>
<tr>
<td><code>XA_DTV_UNICAST_CONNECTION_STATE_CONNECTED_RECEIVING</code></td>
<td>Same as the connected state, but receiving data.</td>
</tr>
<tr>
<td><code>XA_DTV_UNICAST_CONNECTION_STATE_DISCONNECTED</code></td>
<td>Identifies the disconnected state for the unicast bearer.</td>
</tr>
</tbody>
</table>

### 8.2.22. `XA_DTV_UNICAST_TECHNOLOGY`

- `#define XA_DTV_UNICAST_TECHNOLOGY_RTSP ((XAuint32) 0x00000001)`
- `#define XA_DTV_UNICAST_TECHNOLOGY_HTTP ((XAuint32) 0x00000002)`
The macros are defined to identify supported unicast technologies.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XA_DTV_UNICAST_TECHNOLOGY_RTSP</td>
<td>Identifies the RTSP unicast technology.</td>
</tr>
<tr>
<td>XA_DTV_UNICAST_TECHNOLOGY_HTTP</td>
<td>Identifies the HTTP unicast technology.</td>
</tr>
</tbody>
</table>
Part 3: Appendices
## Appendix A: References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMAX AL</td>
<td>The <a href="https://www.khronos.org/openmax">OpenMAX AL 1.1 Specification</a>, Khronos, January 18, 2011</td>
</tr>
<tr>
<td>RFC3376</td>
<td>Internet Group Management Protocol, RFC-3376, IETF, 2002</td>
</tr>
<tr>
<td>MBMS</td>
<td><a href="https://en.wikipedia.org/wiki/Multimedia_Broadcast_and_Multicast_Services">Multimedia Broadcast and Multicast Services (MBMS)</a>, Wikipedia page with links to several 3GPP specifications defining MBMS, 3GPP, 2004-2009</td>
</tr>
<tr>
<td>DVB-H</td>
<td><a href="https://www.etsi.org/deliver/etsi_en/302304/302304v01.11.00_60/en302304v01.11.00">DVB-H specification</a>, ETSI EN 302 304 V1.1.1 (2004-11), ETSI, 2004</td>
</tr>
<tr>
<td>ISDB-T</td>
<td><a href="https://www.arib.or.jp/index-e.html">ARIB Standard for Digital Terrestrial TV Broadcasting</a>, DiBEG, 2005-2008</td>
</tr>
<tr>
<td>IMB</td>
<td><a href="https://www.gsma.com/files/043865bf7b22a3b3fa3c17896266ae82514e5b76.pdf">Integrated Mobile Broadcast - Whitepaper</a>, GSMA, 2009</td>
</tr>
<tr>
<td>DMB</td>
<td><a href="https://www.etsi.org/deliver/etsi_ts/102428/102428v1.2.1_60/ts102428v1.2.1">DMB/DAB specification</a>, ETSI TS 102 428 V1.2.1 (2009-04), ETSI, 2009</td>
</tr>
</tbody>
</table>
## Appendix B: OpenMAX AL Profile-Object Mapping

<table>
<thead>
<tr>
<th>Object</th>
<th>OpenMAX AL Profile</th>
<th>Media Player</th>
<th>Media Player/Recorder</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTVSource</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Player</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program Guide Manager</td>
<td>Mandated</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>Recorder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **Mandated**
- **Optional**
## Appendix C: Object-Interface Mapping

### Legend:

- **Mandated**
- **Optional**

Only the directly used interfaces are shown in the Player and Recorder objects in this table. For a full list of mandated and optional interfaces on these objects, please refer to the OpenMAX AL 1.1 Specification Object-Interface Mapping table.

(1), (2): The recorder object is optional if recording is not supported in the underlying OpenMAX AL implementation. The Recorder object follows the mandating rules from the OpenMAX AL specification. If the Media Player/Recorder Profile is implemented in OpenMAX AL, then the Recorder object is mandated.

(3) Only one of the interfaces marked is mandated. If one of them is implemented, the other ones become optional.

Please refer to section 7 in this specification and the OpenMAX AL 1.1 specification for complete definitions of the interfaces in the table.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Object</th>
<th>DTVSource</th>
<th>Player</th>
<th>Program Guide Manager</th>
<th>Recorder</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAConfigExtensionsItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVContentProtectionItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVPlayerTimedObjectsItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVPlayerTimeShiftControlItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVProgramGuidePurchaseItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVProgramGuideQueryItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVProgramGuideUpdateItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVRecorderModeItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVServiceDataDeliveryItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVServiceInputSelectorItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVSourceBroadcastItf</td>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVSourceLocalItf</td>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVSourceMulticastItf</td>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVSourceUnicastItf</td>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADTVSourceUtilitiesItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADynamicInterfaceManagementItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XADynamicSourceItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XAOBJECTItf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>