AllJoyn™ Media Content Service High-Level Design

Revision 0.2

October 27, 2014

Contents

Figures

Tables

1 Purpose and Scope

1.1 Introduction

The AllJoyn™ Media Delivery framework provides a standardized infrastructure for media applications. It consists of the following components:

- AllJoyn Media Content service—Makes media content available to be browsed and searched.
- AllJoyn Media Control service—Controls the playing of media content.
- Streaming—Enables the media content to be played by the media control.

Together, they allow for rich media experiences across multiple devices.

This document describes only the high-level design for the Media Content service framework.

1.2 References

The following are reference documents:

- *AllJoyn™ Media Control Service Product Requirements Document*, Rev 0.3, October XX, 2014
- *AllJoyn™ About Feature 1.0 Interface Specification*

1.3 Revision history

Table 1 provides the revision history for this document.

Table 1. Revision history

Version	Date	Description
0.1	August 7, 2014	Initial draft
0.2	October 13, 2014	Removed interfaces that are part of the AllJoyn core

1.4 Acronyms

Acronym	Definition
AJAS	AllJoyn Audio Service framework
AJ-MCS	AllJoyn Media Content Service
AJ-MCTRLS	AllJoyn Media Control Service

1.5 Common terms

Term	Definition
AllJoyn Media Delivery Framework	An AllJoyn framework consisting of separate AllJoyn components that together are used for the discovery, distribution and rendering of media content within a network of AllJoyn-enabled devices.
AllJoyn Media Content service	Registers as a service and exposes media-centric files to the AllJoyn network. Includes file type, file metadata, and preferred streaming protocol.
AllJoyn Media Content Agent	A “smart aggregator” that can act as negotiator and mapper between different "media SW stacks.” It is used between Media Delivery Framework components and AllJoyn bridge-supported components for other existing platforms/protocols.
AllJoyn Media Control service	A set of semantic AllJoyn Control interfaces for media controls, targeted at application developers. Controls typically include play, plause, forward, rewind, grouping, channel up/down, volume up/down.
AllJoyn Media application	An application that is created by an application developer for the purpose of discovering media content on AllJoyn devices within an IP-based network and rendering that content to another AllJoyn device on the same network. The application also serves as the controller for media content – play, pause, fast forward, etc.
Content Source	A device that is a content server and stores media content files. A cloud streaming service that provides media content can also be categorized as a Content Source.
Player	A device that is a renderer of media content and can play back media content files (e.g., mobile phones, tablets, televisions, speakers).
PropertyStore	Interface used by an application using the AboutService that maintains the values returned as AboutData.
Streaming protocol	Protocol for streaming media content between AllJoyn-enabled devices.

2 System Design

2.1 Overview

The goal of the Media Content service is to standardize the way content sources are exposed on the AllJoyn network and queried. This document describes the Media Content service interfaces and metadata mappings that comprise this service.

The intent of this chapter's discussion of system design follows.

- The content server must implement the Content Source interface.
- Content sources are MIME type-agnostic. The supported MIME types depends on the content source implementation. To enable this, the Content Source interface exposes the GetContentSourceInfo method. The application logic is responsible to match content source and player capabilities.
- A sample implementation of a content source implementation on Linux is made available. This reference content source generates dummy data for testing.
- An Sample implementation of a basic command line application on Linux to query content sources is made available.

2.2 System architecture

Figure 1 illustrates the Media Deliver framework components

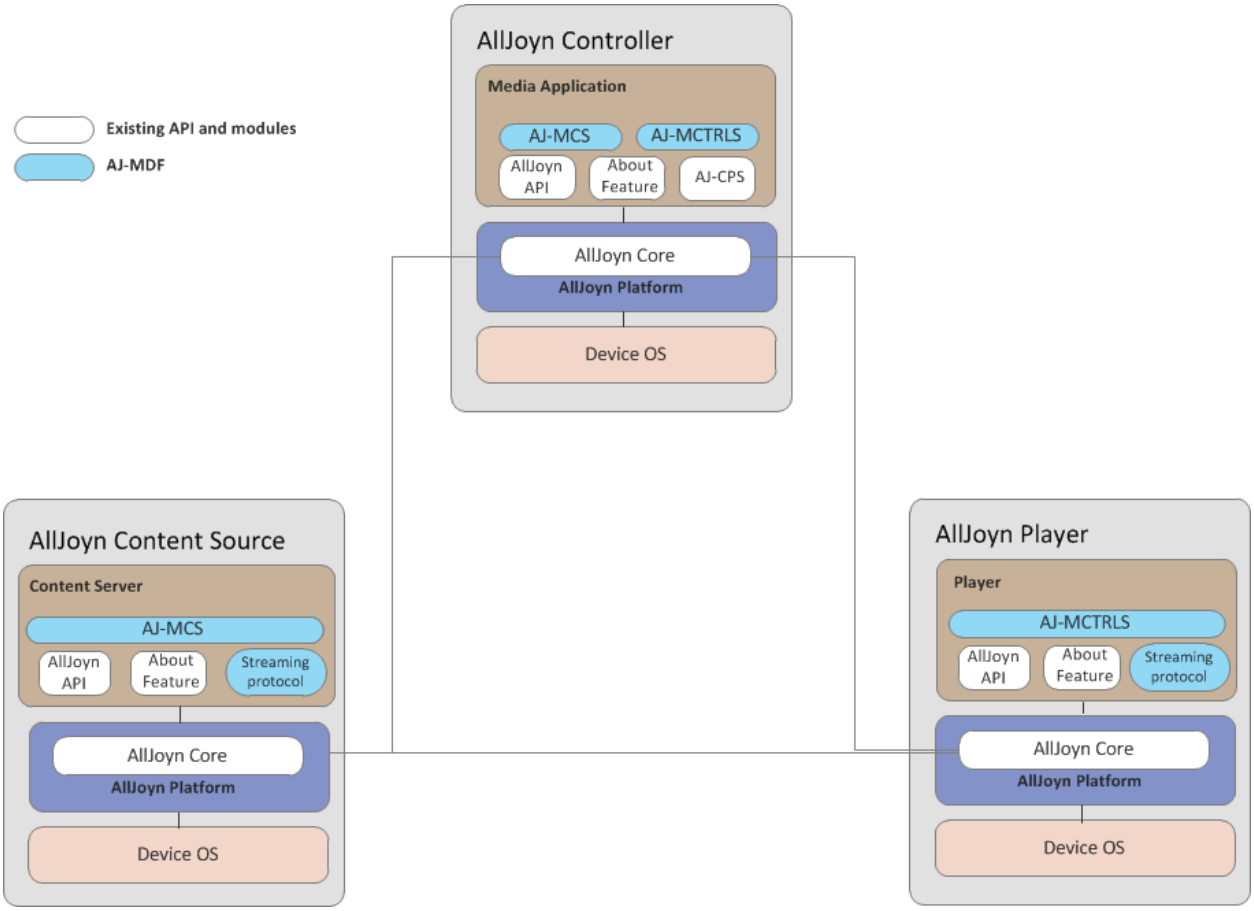

Figure 1. Media Delivery Framework system architecture

2.3 Network layer architecture

Figure 2 illustrates the network layer architecture for the Media Delivery framework. The following subsections define the relevant components.

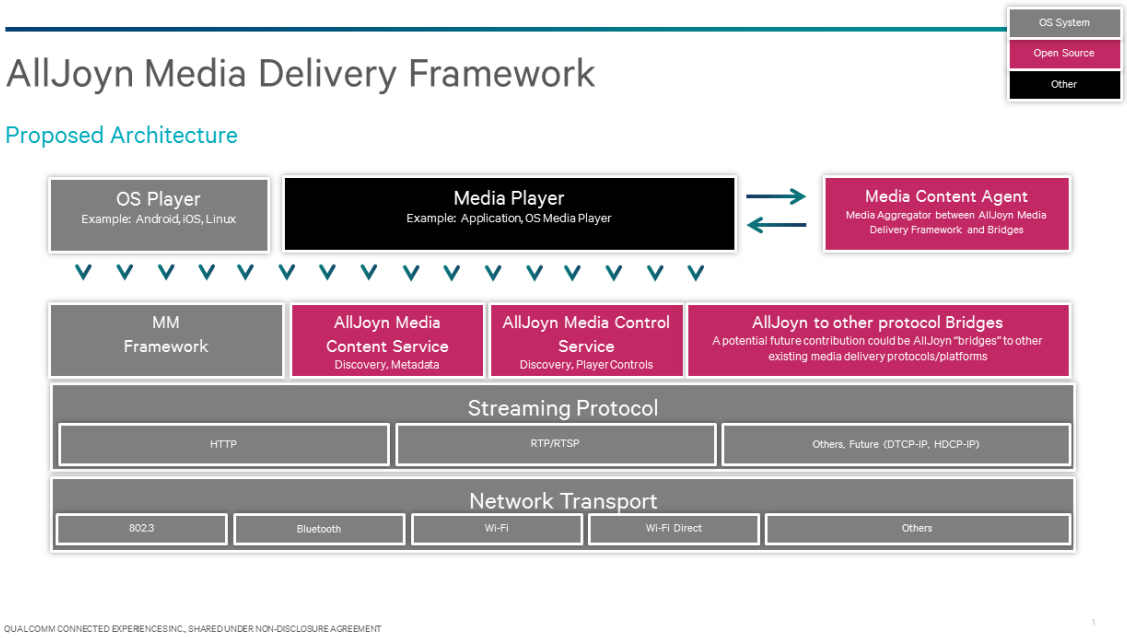

Figure 2. AllJoyn Media Delivery Framework network layer architecture

2.3.1 Media Content service

The Media Content service consists of a content source (server) and a query component that is used to discover content sources.

The content source part of the Media Content service API is used to create and expose a content source on the AllJoyn network, describing its media content's metadata and methods in a standardized format. The query part of the API is used to discover content sources and subsequently query them. It is an implementation for devices that are content servers and for application developers who wants to discovery content sources on the network and query their metadata.

An AllJoyn Media application uses the Media Content service to discover or search for content sources on the network, and retrieve the metadata for media content on the source to present to the end user for selection. It also uses the content source component of the Media Content service API if it wants to expose its own media content on the network.

2.3.2 Media Control service

The Media Control service of a controller and a player. The controller part is used to discover players on the AllJoyn network and to control them i.e., play, pause etc. The player component is used to create a player, describing its methods and capabilities in a standardized way, and advertising it on the network. It is an implementation for devices that are players and for application developers who want to detect players and their playback capabilities on the network.

An AllJoyn Media application uses the Media Control service to discover or search for players on the network and to control them i.e., play, pause etc. It also uses the player component of the Media Control service API if it wants to expose its own player on the network.

2.3.3 Streaming protocol

The Streaming protocol consists of a source (server) and a player. The source is used by the Media Content service API to listen for incoming connection requests from the players and serve the requested content. The player is used by the Media Control service to assemble the packages received from the source before passing the data on to the player.

NOTE

Streaming is not part of the Media Delivery Framework. It is required by the Control application to obtain data from the Content application that is built on top of the Media Delivery Framework. The sample application uses HTTP streaming, but any suitable streaming protocol can be used.

2.4 Discovery via About announcement

The conent services are discovered using the AllJoyn AboutService. For a full description, refer to the *AllJoyn™ About Feature Interface Definition* document.

3 Content Service API

3.1 Overview

The Content Service API enables an application to query content sources for metadata.

It consists of the ContentSource interface.

3.2 ContentSource

3.2.1 Interface name

Interface name	Version	Secured
org.allseen.media.content.ContentSource	1	no

3.2.2 Properties

Property name	Type	List of values	Writable	Description
version	q	N/A	no	Interface version number
displayName	s	N/A	no	Name of the content source.
iconURL	s	N/A	no	URL to the icon for the content service.
lastModified	i	N/A	no	Date the index of the Content Source metadata was last changed. This enabled third parties to cache results.

3.2.3 Methods

The status codes and any error messages returned from method calls will be determined during the implementation of the API and subsequently added to a later version of this document.

Method name	Parameter name	Mandatory	Type	List of values	Direction	Description
Browse	See below					Allow the caller to incrementally explore the content source hierarchy. If authentication is needed before this method can be called, the error org.allseen.error.Unauthorized is raised. The caller of the API will then use the interface org.allseen.media.authentication to authenticate the caller before proceeding with the method call.

Method name	Parameter name	Mandatory	Type	List of values	Direction	Description
	itemID	yes	s	N/A	in	The ID of the item you want to get the children for. Use an item ID of ASCII 0 for the initial browse i.e., to browse the root directory.

	playerCapabilities	yes	a{sv}	N/A	in	Array of key value pairs relating to the player capabilities. These values are used to filter the returned response so that if an item cannot be played on the player the delivery URL is removed and if it can be played the relevant delivery URL is selected. If an empty array is submitted all returned items will have the delivery URLs removed and it is expected that the caller calls GetItem to obtain the delivery URLs for the item.
						Sample capability values:
						`mimeTypes={“audio/mpeg,”video/mp4”}`
						`resolution=2560x1440`
						
						Refer to the *AllJoyn™ Control Service High-Level Design* document for the possible capability values.

	offset	yes	t	N/A	in	Specifies offset within a result. The minimum offset is 0.
	limit	yes	i	N/A	in	Number of results to return. Default is 20.
	sortBy	yes	a(sy)	N/A	in	Criteria for sorting the result. Call GetSortable Fields to find out which fields can be used for sorting the result. `{(“field”, 0|1), (“field”, 0|1),…}` e.g. {(“title”, 0)}. This field can be an empty array if order is not important. ■ 0 – ASC ■ 1 – DESC

Method name	Parameter name	Mandatory	Type	List of values	Direction	Description
	response	yes	(ututa (sssa {sv}))	N/A	out	Browse response. See ModifyOptionalMetadata for details. ■ u – Actual number of results returned. ■ t – Maximum number of results that can be returned. ■ u – Page size / maximum number of results to return e.g., 20. ■ t – Offset within the result. ■ a(sss): ❒ s – itemID ❒ s – title ❒ s – type ■ a{sv} : Arbitrary properties ❒ s – key v – value
GetItem	See below					Call this method with an itemID returned from a browse request to retrieve all the metadata about an item.
	itemId	yes	s	N/A	in	Item ID of the item to fetch the metadata for.
	response	yes	(sssa {sv})	N/A		Item. ■ s – itemID ■ s – title ■ s – type ■ a{sv} : Arbitrary properties ❒ s – key ❒ v – value
GetSortableFields	See below					Return the fields that can be used to sort results returned form this content source.
	sortable	yes	as	N/A	out	Array of fields that can be used to sort the results returned by the content source e.g., artist, album, title, etc.
GetContentSourceInfo	See below for parameter information. See GetContentSourceInfo for mandatory and optional content source information that a media player must return as part of the GetContentSourceInfo method call.					Return information about the content source.
	contentSourceInfo	yes	a{sv}	N/A		Array of key value pairs with information about the content source.
ModifyOptionalMetadata	See below					Lets you modify the optional metadata of an item in the content source.
	itemID	yes	s	N/A	in	ID of the item with the metadata you want to change.

Method name	Parameter name	Mandatory	Type	List of values	Direction	Description
	properties	yes	a{ss}	N/A	in	Any array of key value pairs. The key is the name of the property you want to change and the value the value. The content source should validate this value to ensure it conforms to the relevant format.
Search	See below for parameter information. See Search for additional information about this method.					This method is used to search content sources for content. The GetSortableFields method call is used to find out how the results can be ordered.
	query	yes	s	N/A	in	Free text search string.
	searchMode	yes	q	N/A	in	Search mode to use. ■ 0 – Starts with. The fields must start with the submitted text. 1 – Contains. The fields must contain the submitted text.
	searchType	yes	q	N/A	in	Scope for the free from search i.e., images, videos, or audio. ■ 0 – All ■ 1 – Audio ■ 2 – Images 3 - Video

	playerCapabilities	yes	a{sv}	N/A	in	Array of key value pairs relating to the player capabilities. These values are used to filter the returned response.
						■ If an item cannot be played on the player, the delivery URL is removed.
						■ If it can be played, the relevant delivery URL is selected.
						■ If an empty array is submitted, all returned items have the delivery URLs removed. It is expected that the caller calls GetItem to obtain the delivery URLs for the item.
						Sample capability values:
						`mimeTypes={“audio/mpeg,”video/mp4”}`
						`resolution=2560x1440`
						Refer to the *AllJoyn™ Control Service High-Level Design* document for the possible capability values.

Method name	Parameter name	Mandatory	Type	List of values	Direction	Description
	offset	yes	t	N/A	in	Specifies the offset within a result. The minimum offset is 0.
	limit	yes	i	N/A	in	Number of results to return. Default is 20.
	sortBy	yes	a(sy)	N/A	in	Criteria for sorting the result. Call GetSortableFields to find out which fields can be used for sorting. `{(“field”, 0|1), (“field”, 0|1),…}` e.g., {(“title”, 0)}. This field can be an empty array if order is not important. ■ 0 – ASC ■ 1 – DESC
	response	yes	(ututa (sssa {sv}))	N/A	out	The search response. See ModifyOptionalMetadata for details ■ u – Actual number of results returned. ■ t – Maximum number of results that can be returned. ■ u – Page size / maximum number of results to return e.g., 20. ■ t – Offset within the result. ■ a(sss): ❒ s – itemID ❒ s – title ❒ s - type ■ a{sv} : Arbitrary properties ❒ s – key ❒ v – value

3.2.3.1 GetContentSourceInfo

Table 2 lists the mandatory and optional content source information that a media player must return as part of the GetContentSourceInfo method call.

Table 2. Content source information returned by a media player

Key	Value type	List of values	Mandatory	Description
mimeTypes	as	Standard or non-standard MIME types	yes	Array of the MIME types (string) that the content source supports e.g., {“audio/mpeg”,video/mp4”,”image/jpeg”}

Key	Value type	List of values	Mandatory	Description
transports	aq	■ 1 – HLS ■ 2 – MPEG DASH ■ 3- RTSP ■ 4- MMS ■ 5- RTP ■ 6- RTCP ■ 7- UDP ■ 8- TCP ■ 9- RTMP ■ 10- MPEG-TS ■ 11- RDT ■ 12- WebM	yes	Array of transports that the player supports.
iconURL	string - s	N/A	no	URL to the icon for the content service.

3.2.3.2 Search

If authentication is needed before this method can be called the error org.allseen.error.Unauthorized will be raised. The caller of the API must use the authentication interface to authenticate the caller before proceeding with the method call.

This method performs a free text search over the relevant fields in the content source. The free text argument can be parts of a string e.g., ko in which case any items with ‘ko’ as part of search fields will be returned e.g., ‘koop’.

The search is case insensitive, and the content source implementation specifies which fields to include as part of the searching.

NOTE

The method must return results quickly. Performance is dependent on how much content is indexed and what type of content source it is.

If you search with the below string using the search mode ‘starts with’:

Don’t worry be happy

The results include any items that start with above words in the relevant fields.

If you search with the same string using the search mode ‘contains’, the results include items that have all of the words in one of the relevant fields.

Use quotes to search for a set of words that one of the fields must include, i.e., “Don’t worry be happy”.

3.2.4 Signals

Signal name	Parameters			Sessionless	Description
ContentSourceIndexChanged	**Name**	**Mandatory**	**Signature**	no	Indicates the content source index has changed.
	Timestamp	yes	u		

Signal name	Parameters			Sessionless	Description
MetadataChanged	**Name**	**Mandatory**	**Signature**	no	Indicates the metadata has changed for the specified item. The properties contain the changed key value pairs.
	itemID	yes	s		
	properties	yes	a{sv}		

3.2.5 Introspection XML

```
<node xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="https://www.allseenalliance.org/schemas/in
trospect.xsd">
      <interface name="org.allseen.media.content.ContentSource">
            <property name="version" type="q" access="read" />
            <property name="displayName" type="s" access="read" />
            <property name="iconURL" type="s" access="read" />
            <property name="lastModified" type="i" access="read" />
            <method name="Browse">
                  <arg name="itemID" type="s" direction="in" />
                  <arg name="playerCapabilities" type="a{sv}" direction="in"
/>
                  <arg name="offset" type="t" direction="in" />
                  <arg name="limit" type="i" direction="in" />
                  <arg name="sortBy" type="a(sy)" direction="in" />
                  <!--
                        a(sy):
                              s - field name
                              y - 0 = ASC, 1 DESC
                   -->
                  <arg name="response" type="(ututa(sssa{sv}))"
direction="out" />
                  <!--
                        (utuua(sssa{sv}))
                        u – The number of results returned.
                        t - The maximum number of results that can be
returned.
                        u – The maximum number of results to return e.g. 20
used for paging.
                        t – The offset within the result.
                        a(sss):
                        s - itemID
                        s - title
                        s - type
                        a{sv} : Arbitrary properties
                           s - key
                           v - value
                    -->
            </method>
            <method name="GetItem">
                  <arg name="itemID" type="s" direction="in" />
                  <arg name="response" type="(sssa{sv})" direction="out" />
```

```
                        <!--
                               s - itemID
                               s - title
                               s - type
                               a{sv} : Arbitrary properties
                                  s - key
                                  v - value
                         -->
                </method>
                <method name="GetSortableFields">
                       <arg name="sortableFields" type="as" direction="out" />
                </method>
                <method name="GetContentSourceInfo">
                <arg name="supportedMIMETypes" type="a{sv}" direction="out" />
                </method>
                <method name="ModifyOptionalMetaData">
                       <arg name="itemID" type="s" direction="in" />
                       <arg name="properties" type="a{ss}" direction="in" />
                </method>
                <method name="Search">
                       <arg name="query" type="s" direction="in" />
                       <arg name="searchMode" type="q" direction="in" />
                       <!--
                              searchMode: 0 = Starts with, 1 = contains
                        -->
                       <arg name="searchType" type="q" direction="in" />
                       <!--
                              searchType: 0 = All items, 1 =  Audio items,
                                           2 = Image items, 3 = Video items
                        -->
                       <arg name="playerCapabilities" type="a{sv}" direction="in"
/>
                       <arg name="offset" type="t" direction="in" />
                       <arg name="limit" type="i" direction="in" />
                       <arg name="sortBy" type="a(sy)" direction="in" />
                       <arg name="response" type="(ututa(sssa{sv}))"
direction="out" />
                       <!-- See browse -->
                </method>
                <signal name="ContentSourceIndexChanged">
                       <arg name="timeStamp" type="i" />
                </signal>
                <signal name="MetadataChanged">
                       <arg name="itemID" type="s" />
                       <arg name="properties" type="a{sv}" />
                </signal>
                </interface>
</node>
```

4 Error handling

The method calls in the interfaces will use the AllJoyn error message handling (the ER_BUS_REPLY_IS_ERROR_MESSAGE feature) to set the error name and error message.

Table 3 lists the possible errors raised by the content interfaces.

Table 3. ContentSource interface error handling

Error name	Error message
org.allseen.error.OutOfRange	Value out of range
org.allseen.error.InvalidValue	Invalid value
org.allseen.error.InvalidProperty	Invalid property
org.allseen.error.MethodNotAllowed	Method call not allowed
org.allseen.error.Unauthorized	Unauthorized

4.1 BusObject map

Figure 3 shows the basic organization of AllJoyn objects used to register content sources.

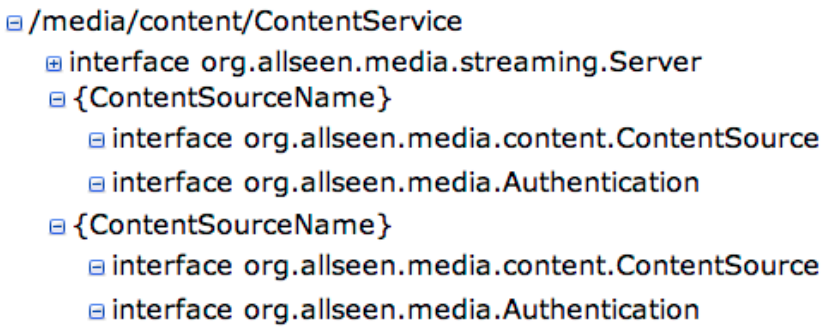

Figure 3. BusObject map

5 Media Content Metadata

5.1 Mandatory metadata

<table>
<tr><th>Key</th><th>Type</th><th colspan="3">List of values</th><th>Description</th></tr>
<tr><td>itemID</td><td>s</td><td colspan="3">N/A</td><td>An unique identifier for the item</td></tr>
<tr><td>title</td><td>s</td><td colspan="3">N/A</td><td>The name of the media item</td></tr>
<tr><td rowspan="20">type</td><td rowspan="20">s</td><td>Type</td><td>Value</td><td>Description</td><td rowspan="20"></td></tr>
<tr><td rowspan="5">audio</td><td>media.audio</td><td>Generic type for audio.</td></tr>
<tr><td>media.audio.book</td><td>Audio book.</td></tr>
<tr><td>media.audio.broadcast</td><td>Audio broadcast.</td></tr>
<tr><td>media.audio.podcast</td><td>Audio podcast.</td></tr>
<tr><td>media.audio.track</td><td>Audio track.</td></tr>
<tr><td>image</td><td>media.image</td><td>Generic type for images.</td></tr>
<tr><td rowspan="5">video</td><td>media.video</td><td>Generic type for videos.</td></tr>
<tr><td>media.video.broadcast</td><td>Video broadcast.</td></tr>
<tr><td>media.video.clip</td><td>Video clip.</td></tr>
<tr><td>media.video.movie</td><td>Movie.</td></tr>
<tr><td>media.video.podcast</td><td>Video podcast</td></tr>
<tr><td rowspan="9">container</td><td>container</td><td>Generic type used for grouping of items.</td></tr>
<tr><td>container.album</td><td>Generic type for albums.</td></tr>
<tr><td>container.album.audio</td><td>Music album.</td></tr>
<tr><td>container.album.image</td><td>Photo album.</td></tr>
<tr><td>container.artist</td><td>Artist.</td></tr>
<tr><td>container.genre.video</td><td>Movie genre.</td></tr>
<tr><td>container.genre.audio</td><td>Music genre</td></tr>
<tr><td>container.playlist</td><td>Playlist.</td></tr>
</table>

5.1.1 Media items

The media items listed below se are required to be able to play an item.

Key	Signature	List of values	Description
Item-> deliveryMethods[X] -> url	s	N/A	The location of the media.
Item-> deliveryMethods[X] -> mimeType	s	N/A	The MIME type of the media.

5.2 Optional metadata

The optional metadata items listed below are used to describe the item content.

Key	Signature	List of values	Description
additionalInfo	s	N/A	Generic field for specifying additional information about the media.
album	s	N/A	Album title
albumArtist	s	N/A	Album artist
albumRating	y	0-5	“Star” rating
albumURL	s	N/A	Album url
artist	s	N/A	Artist that performed the item
artistURI	s	N/A	Artist URL
bitrate	q	N/A	Bitrate in bytes per second
bpm	g	N/A	Beats per minute
bitsPerSample	q	N/A	Bits per sample
colorDepth	q	N/A	Color depth in bits of the resources e.g., 24 for true color.
compilation	b	N/A	Indicates if item is part of a compilation
composer	s	N/A	Composer of the item
contributor	s	N/A	Entity responsible for making contributions to the content.
creator		N/A	Entity responsible for making the content of the resource.
creationDate	s	Date	Date the item was created. See Date value information.
discCount	g	N/A	Total number of discs in a multi-disc release
discNumber	g	N/A	Disc number in a multi-disc release
description	s	N/A	Description of the content
displayName	s	N/A	The display name if available.
duration	u	N/A	Duration of the item in milliseconds
framesPerSecond	i	N/A	The number of frames per second (fps).
genre	s	N/A	Genre of the item
iconSmallURL	s	N/A	The URL of a small asset. Size: 128x128

Key	Signature	List of values	Description
iconMediumURL	s	N/A	URL of a medium asset. Size: 256x256
iconLargeURL	s	N/A	URL of a large asset. Size: 600x600
iconHDURL	s	N/A	URL of a HD asset. Size: 1024x1024
isExplicit	b	N/A	Indicates whether the item is explicit
isFamilyFriendly	b	N/A	Indicates whether the item is family friendly
language	s	N/A	language of the content specified in the ISO 639-2 format e.g., swe.
mimeType	s	N/A	Item's MIME type.
modificationDate	s	Date	Date the item was last modified. See Date value information.
nrAudioChannels	y	N/A	Number of audio channels
numberOfChildren	q	N/A	Number direct children of a folder. Only applies to folders.
parentId	s	N/A	Parent ID of the item.
priority	i	N/A	Priority associated with the deliver method e.g., ajmsp = 1 and http = 2. The lower the number the higher priority.
publisher	s	N/A	Publisher of the creative work.
rating	y	0-5	"Star" rating
regionsAllowed	s	N/A	Comma-separated string of the regions where the media is allowed. If not specified, then it's assumed to be allowed everywhere. Specify the countries in ISO 3166 format.
releaseDate	s	Date	Date the item was first released. See Date value information.
resolution	s	N/A	X*Y resolution. One or more digits followed by 'x' followed by more digits i.e., 2560x1440
rights	s	N/A	Information about the rights of the content i.e., copyright.
sampleFrequency	q	N/A	Sample frequency in Hz
size	u	N/A	Size in bytes of the resource.
subject	s	N/A	Subject of the content.
trackCount	g	N/A	Total number of tracks on the album
trackNumber	g	N/A	Track number of the item within the album
tags	s	N/A	Comma-separated list of tags associated with the content.

5.2.1 Date value information

Date values are to only be represented as strings in one of the following ISO 8601 formats:

- Year only: “<year>” where <year> includes all digits of the year.
- Date: “<year><month><day>” where:
  - <year> includes all digits of the year
  - <month> is a 2-digit representation of month (“01” = January)
  - <day> is the day of the month (e.g., “09”).
- Date and time: “<year><month><day>T<hour><minute><second>±<offset>” where:
  - <year>, <month>, and <day> are the same as the date previously specified
  - <hour> is the hour (“00” through “23”)
  - <minute> is the minute (“00” through “59”)
  - <second> is the second (“00” through “59”)
  - <offset> is the offset from UTC (“-0800” corresponds to Pacific Standard Time)
- Date and time: “<year><month><day>T<hour><minute>±<offset>” is the same as the previous date and time specification except without the <second> (seconds) portion.

5.3 Response types

This section specifies the metadata associated with the various response types. Types that are typically only associated with the mandatory metadata are not detailed below.

5.3.1 All responses

Property	Mandatory
itemId	true
title	true
type	true

5.3.2 Album (type=container.album.music)

Property	Mandatory
albumURL	false
artistID	false
artist	false
artistURL	false
genre	false

5.3.3 Album (type=container.album.photo)

Property	Mandatory
date	false
location	false

5.3.4 Artist (type=container.artist)

Property	Mandatory
artistURL	false
genre	false

5.3.5 Playlst (type=container.playlist)

Property	Mandatory	Description
URL	true	URL to an external playlist file (mp3u).
Artist	false	Names of the artists on the playlist.
genre	false	Applies to the entire playlist.
description	false	Playlist description.
date	false	Creation date.

5.3.6 Audio item (type=media.audio)

Object	Property	Mandatory
item	additionalInfo	false
	album	false
	albumArtist	false
	albumRating	false
	albumURL	false
	artist	false
	artistURI	false
	compilation	false
	composer	false
	contributor	false
	creator	false
	creationDate	false
	discCount	false
	discNumber	false
	description	false
	displayName	false
	duration	false

Object	Property	Mandatory
	genre	false
	iconURL	false
	isExplicit	false
	isFamilyFriendly	false
	language	false
	modificationDate	false
	publisher	false
	rating	false
	regionsAllowed	false
	releaseDate	false
	rights	false
	subject	false
	trackCount	false
	trackNumber	false
	tags	false
Item > deliveryMethods	additionalInfo	false
	bitrate	false
	bpm	false
	bitsPerSampe	false
	mimeType	true
	priority	false
	nrAudioChannels	false
	sampleFrequency	false
	size	false
	url	true

5.3.7 Image item (type=media.image)

Object	Property	Mandatory
item	additionalInfo	false
	album	false
	compilation	false
	composer	false
	contributor	false
	creationDate	false
	creator	false
	description	false
	displayName	false
	iconURL	false
	isExplicit	false
	isFamilyFriendly	false

Object	Property	Mandatory
	language	false
	modificationDate	false
	publisher	false
	rating	false
	regionsAllowed	false
	releaseDate	false
	rights	false
	subject	false
	tags	false
Item > deliveryMethods	additionalInfo	false
	colorDepth	false
	mimeType	true
	priority	false
	resolution	false
	size	false
	url	true

5.3.8 Video item (type=media.video)

Object	Property	Mandatory
item	additionalInfo	false
	contributor	false
	compilation	false
	composer	false
	creationDate	false
	creator	false
	description	false
	discCount	false
	discNumber	false
	displayName	false
	duration	false
	genre	false
	iconURL	false
	isExplicit	false
	isFamilyFriendly	false
	language	false
	modificationDate	false
	publisher	false
	rating	false
	regionsAllowed	false
	releaseDate	false

Object	Property	Mandatory
	rights	false
	subject	false
	tags	false
	trackCount	false
	trackNumber	false
Item > deliveryMethods	additionalInfo	false
	framesPerSecond	false
	mimeType	true
	priority	false
	resolution	false
	size	false
	url	true

5.4 Extending metadata

The AdditionalInfo field sends custom metadata, but an application can also have their own custom metadata fields. These fields will be unknown to standard AllJoyn players.

5.5 Dynamic metadata support

To broadcast changing metadata during continuous streaming, e.g., a radio station, the content source emits the signal MetadataChanged with the relevant metadata.

5.6 Editing optional metadata

During playback, a user can rate the content or change some of the optional metadata listed in Optional metadata. To do this, the content source must implement ModifyOptionalMetaData (See ContentSource for details).