HbbTV® Specification Version 1.5

1st August 2012

Contents

1 Scope

The present document extends the HbbTV specification to address a number of immediate requirements which will clearly be addressed at a national level if they are not promptly addressed at a European level. The principal requirements of this sort are HTTP adaptive streaming, a common encryption scheme allowing the use of multiple DRM technologies and access to DVB-SI EIT schedule information.

The present document has been integrated with the text of ETSI TS 102 796 V1.1.1 and the errata to that document and proposed to ETSI as version 1.2.1 of TS 102 796. For avoidance of doubt, the contents of this document have not been reviewed or approved by ETSI and may be changed as part of that process.

This is a temporary document that will have no meaning once ETSI publish TS 102 796 V1.2.1. This document should not be referenced except in documents that will be updated as necessary to reference ETSI TS 102 796 V1.2.1 once that is published.

Test cases will not be developed referencing this document. Test cases for HbbTV receivers implementing the new features described in this document will reference TS 102 796 V1.2.1.

2 References

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 102 796 (V1.1.1): "Hybrid Broadcast Broadband TV".

[2] ETSI EN 300 468 (V1.10.1): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[3] ISO/IEC 23009-1 : Information technology –Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats

[4] ISO/IEC 23001-7 : Information technology -- MPEG systems technologies -- Part 7: Common encryption in ISO base media file format files

NOTE: This was previously Annex I of [5]

[5] ISO/IEC 14496-12 ISO Base File Format

[6] Open IPTV Forum Release 1 specification, Volume 5 (V1.2): "Declarative Application Environment".

[7] Open IPTV Forum Release 2 specification, Volume 2 (V1.2): "Media Formats".

NOTE: Available at http://www.oipf.tv/downloads.html.

[8] ETSI TS 101 154 (V1.10.1): “Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream”

[9] ISO/IEC 14496-12:2008/FDAM 3:2011(E): “Information technology — Coding of audio-visual objects — Part 12: ISO base media file format, AMENDMENT 3: DASH support and RTP reception hint track processing”

[10] ETSI TS 102 822-3-1 V1.7.1 (2011-11): “Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime");Part 3: Metadata;Sub-part 1: Phase 1 - Metadata schemas”

[11] ETSI TS 102 366 V1.2.1 (2008-08): “Digital Audio Compression (AC-3, Enhanced AC-3) Standard”

[12] W3C Recommendation “Exclusive XML Canonicalization Specification Version 1.0”, 18 July 2002
NOTE: Available at http://www.w3.org/TR/xml-exc-c14n/

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 102 796 [1] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 102 796 [1] and the following apply:

AES	Advanced Encryption Standard
AVC	Advanced Video Coding
BFF	Base File Format
CTR	Counter
DASH	Dynamic Adaptive Streaming over HTTP
DVB-SI	DVB Service Information
ISO	International Organization for Standardization
ISOBMFF	ISO Base Media File Format
KID	Key Identifier
MPD	Media Presentation Description

4 Overview

4.1 Relationship with TS 102 796

The present document is an extension to the Hybrid Broadcast Broadband TV specification [1] and full compliance with that document is required.

4.2 Specification Overview

The present document largely references parts of already available standards and specifications and adapts those parts where necessary. The most significant documents directly referenced from the present document are the following:

- ETSI TS 102 796 V 1.1.1 “Hybrid Broadcast Broadband TV" [1].
- Open IPTV Forum Release 1 Volume 5 - Declarative Application Environment of the Open IPTV Forum [6].
- MPEG DASH – formally known as ISO/IEC 23009-1 : Information technology –Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats [3].
- MPEG CENC – formally known as ISO/IEC 23001-7 : Information technology -- MPEG systems technologies -- Part 7: Common encryption in ISO base media file format files[4].

Figure 1 shows a graphical overview of the relationship between the contents of the present document and the above mentioned specifications.

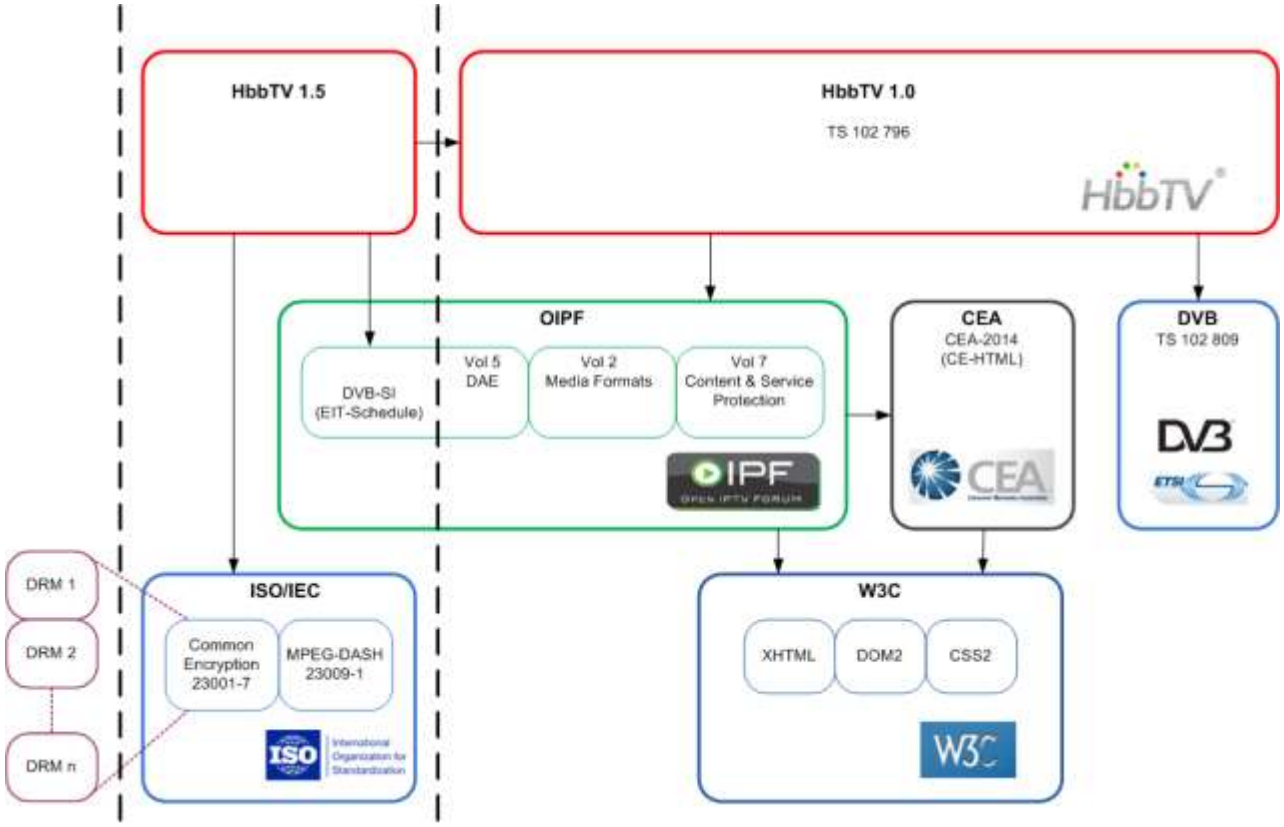

Figure 1: Specification Overview

5 Void

This clause is intentionally left blank.

6 Void

This clause is intentionally left blank .

7 Formats and protocols

7.1 Void

This clause is intentionally left blank .

7.2 Broadcast-specific formats and protocols

7.2.1 Signalling of Applications

In table 5 of TS 102 796 [1], "Supported application signalling features", the text in the Notes column for the row for section 5.2.5, "platform profiles" shall be revised as follows:

The version.minor field shall be changed to '2'.

The following text shall be added to the end of the existing text: "The 3 most significant bits of the `application_profile` are reserved for future use."

7.3 Broadband-specific formats and protocols

7.3.1 HTTP Adaptive Streaming

HTTP adaptive streaming shall be supported using MPEG DASH as defined in annex B.

7.3.2 HTTP User-Agent Header

The HTTP User-Agent header shall be as in TS 102 796 [1] with “HbbTV/1.1.1” replaced with “HbbTV/1.2.1”.

8 Void

This clause intentionally left blank.

9 System Integration

9.1 Presentation of adaptive bitrate content

Terminals shall support applications setting the data attribute of a CEA-2014 A/V control object to a URL referencing an MPD as defined in DASH [3] and identified by the MIME type in annex C of that document. The type attribute of the A/V object shall be set to "application/dash+xml".

NOTE: This is an intentional deviation from requirement 5.7.1.a of CEA-2014[] where the type attribute contains the type of the video or audio.

In order to play the content, the terminal shall fetch the MPD from the URL, interpret the MPD and select an initial set of representations. If at any time the MPD is found to be not valid according to the XML schema or semantics defined in DASH[3], the A/V control object shall go to play state 6 (‘error’) with error value 4 (‘content corrupt or invalid’).

If the content access streaming descriptor defined in annex E.2 of the OIPF DAE specification [6] is supported then terminals shall support the <ContentURL> element of this descriptor referencing an MPD as defined in DASH [3]. In these circumstances, the other requirements from the preceeding paragraph shall apply.

If the terminal supports trick modes, the behaviour defined in clause A.2.3 shall be supported including the generation of a PlaySpeedChangedd event reporting the actual speed of fast forwards and fast rewind.

10 Capabilities

10.1 Void

This clause intentionally left blank.

10.2 Void

This clause intentionally left blank.

10.3 XML Capabilities

The xmlCapabilities property of the application/oipfCapabilities object (as defined in clause 7.15.3 of the OIPF DAE specification [6]) shall include the following;

- The <clientMetadata> element shall include the <dvb-si> metadata system/protocol name as defined in section 9.3.7 of the OIPF DAE specification [6].
- Support for HTTP adaptive streaming shall be indicated by including <video_profile> elements including “dash” as the transport attribute.

NOTE: There are currently no <audio_profile> elements defined which include 'dash' as the transport attribute.

The XML document returned by the xmlCapabilities property of the application/oipfCapabilities embedded object for the base level of features (see clause 10.2.4 of TS 102 796 [1]) shall describe an XML document that when canonicalized according to the W3C XML Canonicalization specification [12] shall be equal to the canonicalized form of the following XML;

```
<profilelist>Redm
<ui_profile name="OITF_HD_UIPROF+DVB_S+TRICK_MODE" >
 <ext>
   <parentalcontrol schemes="dvb-si">true</parentalcontrol>
 </ext>
 </ui_profile>

<clientMetadata type="dvb-si">true</clientMetadata>
<video_profile name="TS_AVC_SD_25_HEAAC" type="video/mpeg" />
<video_profile name="TS_AVC_HD_25_HEAAC" type="video/mpeg" />
<video_profile name="MP4_AVC_SD_25_HEAAC" type="video/mp4" />
<video_profile name="MP4_AVC_HD_25_HEAAC" type="video/mp4" />
<video_profile name="MP4_AVC_SD_25_HEAAC" type="video/mp4" transport="dash"/>
<video_profile name="MP4_AVC_HD_25_HEAAC" type="video/mp4" transport="dash"/>
<audio_profile name="MPEG1_L3" type="audio/mpeg"/>
<audio_profile name="HEAAC" type="audio/mp4"/>
</profilelist>
```

As specified in clause 7.3.1.1 of TS 102 796 [1], terminals shall support Enhanced-AC3 for the broadband connection when it is supported by the broadcast connection. The format label “E-AC3” (as defined in clause 8.1.3 of the OIPF Release 2 Media Formats specification [13]) shall be used to indicate the support of Enhanced AC-3 by the terminal.

Where E-AC3 is applicable, the XML document returned by the xmlCapabilities property of the application/oipfCapabilities embedded object for the base level of features (see clause 10.2.4 of TS 102 796 [1]) shall describe an XML document that when canonicalized according to the W3C XML Canonicalization specification [12] shall be equal to the canonicalized form of the following XML;

```
<profilelist>
<ui_profile name="OITF_HD_UIPROF+DVB_S+TRICK_MODE" >
<ext>
   <parentalcontrol schemes="dvb-si">true</parentalcontrol>
</ext>
</ui_profile>
<clientMetadata type= "dvb-si">true</clientMetadata>
<video_profile name= "TS_AVC_SD_25_E-AC3" type="video/mpeg" />
<video_profile name="TS_AVC_HD_25_E-AC3" type="video/mpeg" />
<video_profile name="TS_AVC_SD_25_HEAAC" type="video/mpeg" />
<video_profile name="TS_AVC_HD_25_HEAAC" type="video/mpeg" />
<video_profile name="MP4_AVC_SD_25_E-AC3" type=”video/mp4" />
<video_profile name="MP4_AVC_HD_25_E-AC3" type=”video/mp4" />
<video_profile name="MP4_AVC_SD_25_HEAAC" type="video/mp4" />
<video_profile name="MP4_AVC_HD_25_HEAAC" type="video/mp4" />
<video_profile name="MP4_AVC_SD_25_E-AC3" type="video/mp4" transport="dash"/>
<video_profile name="MP4_AVC_HD_25_E-AC3" type="video/mp4" transport="dash"/>
<video_profile name="MP4_AVC_SD_25_HEAAC" type="video/mp4" transport="dash"/>
<video_profile name="MP4_AVC_HD_25_HEAAC" type="video/mp4" transport="dash"/>
<audio_profile name="MPEG1_L3" type="audio/mpeg"/>
<audio_profile name="HEAAC" type="audio/mp4"/>
</profilelist>
```

The modifications to the corresponding XML text defined in clause 10.2.4 of TS 102 796 [1] also apply to this XML text. For example, “DVB_S” can be replaced by the appropriate string(s) for the supported broadcast delivery system(s).

11 Security

11.1 Protected content via Broadband

Support for delivering protected content via the broadband channel is optional in the present document. If this is supported and the content is provided in an ISO base media file format, then one mechanism by which the content may be encrypted is MPEG common encryption as defined by CENC [4] and constrained by Annex C of the present document.

Annex A – OIPF DAE Specification Profile

A.1 Access to EIT Schedule Information

Access to DVB-SI EIT schedule information (as defined in EN 300 468 [2]) shall be supported with the following sections and sub-sections of the OIPF DAE specification [6]. This replaces the corresponding rows of Table A.1: "Section-by-section profile of the OIPF DAE specification" in the HbbTV specification [1].

Table A.1: Amended section-by-section profile of the OIPF DAE specification

Section, sub-section	Reference in DAE [6]	Status in the present document	Notes	Security
The application/oipfSearchManager embedded object	7.12.1	M(*)as modified by clause A.2.9.1 of the present document	The `guideDaysAvailable` and `onMetadataUpdate` properties are not included. For the `createSearch` method, only the value '1' of the `searchTarget` parameter is included. For the function "`onMetadataSearch`", the value "2" for the `state` property is not supported.	Broadcast-related
The MetadataSearch class	7.12.2	M(*)as modified by clause A.2.9.2 of the present document	Only the value '1' of the `searchTarget` property is included. For the `createQuery` method, only the following case-insensitive values for the `field` parameter are included - "Programme.startTime", "Programme.name", "Programme.programmeID". These shall correspond to the properties of the same name. The value "7" for the `comparison` property is not supported. The `addRatingConstraint`, `addCurrentRatingConstraint` and addChannelConstraint(ChannelList) methods are not included. The `orderBy` method is not included – all search results shall be returned ordered first by channel, in the same order as presented to applications through a ChannelList object, then by start time in ascending order.	Broadcast-related
The Query class	7.12.3	M(*) as modified by clause A.2.9.3 of the present document	The values of the `field` and `comparison` properties are constrained as defined above for the parameters of the same name on the `createQuery` method of the MetadataSearch class.	Broadcast-related
The SearchResults class	7.12.4	M(*) as modified by clause A.2.9.4 of the present document		Broadcast-related
The MetadataSearchEvent class	7.12.5	NI		
The MetadataUpdateEvent class	7.12.6	NI		

The programme class				
Basics	7.16.2.1, 7.16.2.2	M(*)	The following properties are required: - name - programmeID - programmeIDType - description - longDescription - startTime - duration - channelID - parentalRating The constants defined in clause 7.16.2.1 shall be supported however support for CRIDs is outside the scope of the present document. The following method is required for Programme objects returned by the programmes property of the video/broadcast object: - getSIDescriptors All other properties and methods are not included.	Broadcast-related

The meanings of the Status and Security columns shall be as defined in tables A.2 “Table A.2: Key to security column” and A.3 “Table A.3: Key to status column” of the HbbTV specification [1].

A.2 Modifications, Extensions and Clarifications

A.2.1 Metadata Property Mappings

A.2.1.1 The Channel class

This clause is replaced by the definition within clause 8.4.3 of the OIPF DAE specification [6] for channels of type ID_DVB_*.

A.2.1.2 The Programme class and related classes

This clause is replaced by clause 8.4.4 of the OIPF DAE specification [6] for those properties that are included in the present document.

A.2.1.2 The AVComponent class

When media content components are delivered using DASH, instances of the AVComponent class shall refer to AdaptationSets carrying audio, video or subtitles.

When an instance of the `AVComponent` class refers to a DASH audio media content component which is identified as being audio description (as defined in clause B.2.4 Role Related Requirementsbelow), the `audioDescription` property of the `AVComponent` shall be set to true.

When an instance of the `AVComponent` class refers to a DASH audio media content component, the `language` property shall be set from value of the lang attribute in the MPD – whether set explicitly for that component or inherited. If the `lang` attribute in the MPD is not set for a media content component then the value of the `language` property in the corresponding AVComponent class shall be `Undefined`. The contents of the language field in the media header “mdhd” of the track shall be ignored.

When an instance of the AVComponent class refers to a DASH media content component, the componentTag shall be the value of the id attribute on the Adaptation Set (if provided).

A.2.2 Extension to the Channel class

This clause is replaced by the definition of the nid property in clause 7.13.11.2 of the OIPF DAE specification [6].

A.2.3 play speed approximation

This clause is replaced by changes in annex B of the OIPF DAE specification [6].

A.2.4 DOCTYPE

All XHTML documents in an HbbTV application shall include a DOCTYPE as follows;

- One of the 3 DOCTYPEs defined in clause A.2.6.2 of HbbTV [1].
- The following "doctype" declaration:
  <!DOCTYPE html PUBLIC "-//HbbTV//1.2.1//EN" "http://www.hbbtv.org/dtd/HbbTV-1.2.1.dtd">

NOTE: Terminals implementing HbbTV [1] may reject documents with the 1.2.1 doctype. Hence this doctype shall only be used for applications which are so dependent on features in the present document that it would be meaningless for a 1.1.1 terminal to even start them.

A.2.5 Void

This section intentionally left blank.

A.2.6 Void

This section intentionally left blank.

A.2.7 Void

This section intentionally left blank.

A.2.8 Void

This section intentionally left blank.

A.2.9 Metadata APIs

A.2.9.1 The application/oipfSearchManager embedded object

A.2.9.1.1 Introduction

Terminals SHALL implement the "`application/oipfSearchManager`" embedded object. This object provides a mechanism for applications to create and manage metadata searches.

The following example shows how a metadata search can be constructed and executed:

```
// Event handler function for asynchronous search results
function handleSearchResults(search, status) {
  if (status == 0) {
    // our search has finished


    // do stuff with the results
    var myResult = search.result[0];

  }
}

// Function that creates and starts a search
function doSearch() {

  // create a new search for on-demand content
```

```
  mySearchManager = document.getElementById("searchManager");
  mySearch = mySearchmanager.createSearch(1);

  // search for any programme with "space" in the title as a word
  // or part of a word
  myQuery = mySearch.createQuery(
      "Programme.name",
      6,
      "space");
  mySearch.setQuery(myQuery);


  mySearchManager.onMetadataSearch = handleSearchResults;

  mySearch.result.getResults(0, 10);
}
```

A.2.9.1.2 Properties

`function onMetadataUpdate( Integer action, Integer info, Object object )`

This function is the DOM 0 event handler for events indicating changes in metadata. This SHALL be raised when a new version of the metadata is discovered. Note that new versions of metadata can be made available without any of the individual items of metadata changing. It is an application's responsibility to determine what, if anything, has changed.

The specified function is called with the arguments action, info and object. These arguments are defined as follows:

- `Integer action` – the type of update that has taken place. This field will take one of the following values:

Value	Description
1	A new version of metadata is available and applications SHOULD discard all references to Programme objects immediately and re-acquire them.

- `Integer info` – this argument SHALL take the value undefined.
- `Object object` – this argument SHALL take the value undefined.

`function onMetadataSearch( MetadataSearch search, Integer status )`

This function is the DOM 0 event handler for events relating to metadata searches. The specified function is called with the arguments search and state. These arguments are defined as follows:

- `MetadataSearch search` – the affected search
- `Number status` – the new status of the search

Value	Description
0	Search has finished. This event SHALL be generated when a search has completed.
1	Not included in the present document.
2	Not included in the present document.
3	The search has been aborted, either because of a call to `SearchResults.abort()` or because the parameters for the search have been modified (e.g. the query, constraints or search target).
4	The search cannot be completed due to a lack of resources or any other reason (e.g. insufficient memory is available to cache all of the requested results).

For the intrinsic events "`onMetadataSearch`" and "`onMetadataUpdate`", corresponding DOM level 2 events SHALL be generated, in the following manner:

Intrinsic event	Corresponding DOM 2 event	DOM 2 Event properties
`onMetadataSearch`	`MetadataSearch`	Bubbles: No Cancelable: No Context Info: search, status
`onMetadataUpdate`	`MetadataUpdate`	Bubbles: No Cancelable: No Context Info: action, info, object

These events are targeted at the `application/oipfSearchManager` object.

A.2.9.1.3 Methods

MetadataSearch createSearch(Integer searchTarget)		
Description	Create a MetadataSearch object that can be used to search the metadata.	
Arguments	*searchTarget*	The metadata that should be searched. Values other than 1 SHALL cause the search to return no results.

ChannelConfig getChannelConfig()	
Description	Returns the channel line-up of the tuner in the form of a ChannelConfig object as defined in section 7.13.9 of [6]. This includes the favourite lists. The ChannelConfig object returned from this function SHALL be identical to the ChannelConfig object returned from the getChannelConfig() method on the video/broadcast object as defined in section 7.13.1.3 of the DAE specification [6**Error! Reference source not found.**].

A.2.9.2 The MetadataSearch class

A.2.9.2.1 Introduction

A MetadataSearch object represents a query of the metadata about available programmes. Applications can create MetadataSearch objects using the createSearch() method on the application/oipfSearchManager object.

Each search consists of three steps:

1) Definition of the query. The application creates a MetadataSearch object, creates its associated Query object and sets any applicable constraints and result ordering.

2) Acquisition of results. The receiver acquires some or all of the items that match the specified query and constraints, and caches the requested subset of the results. This is triggered by a call to getResults().

3) Retrieval. The application accesses the results via the SearchResults class.

The MetadataSearch and SearchResults classes work together to manage an individual search. For every search, the MetadataSearch object and its corresponding SearchResults object SHALL be in one of three states as described in table 1. Figure A.1 below shows the transitions between these states:

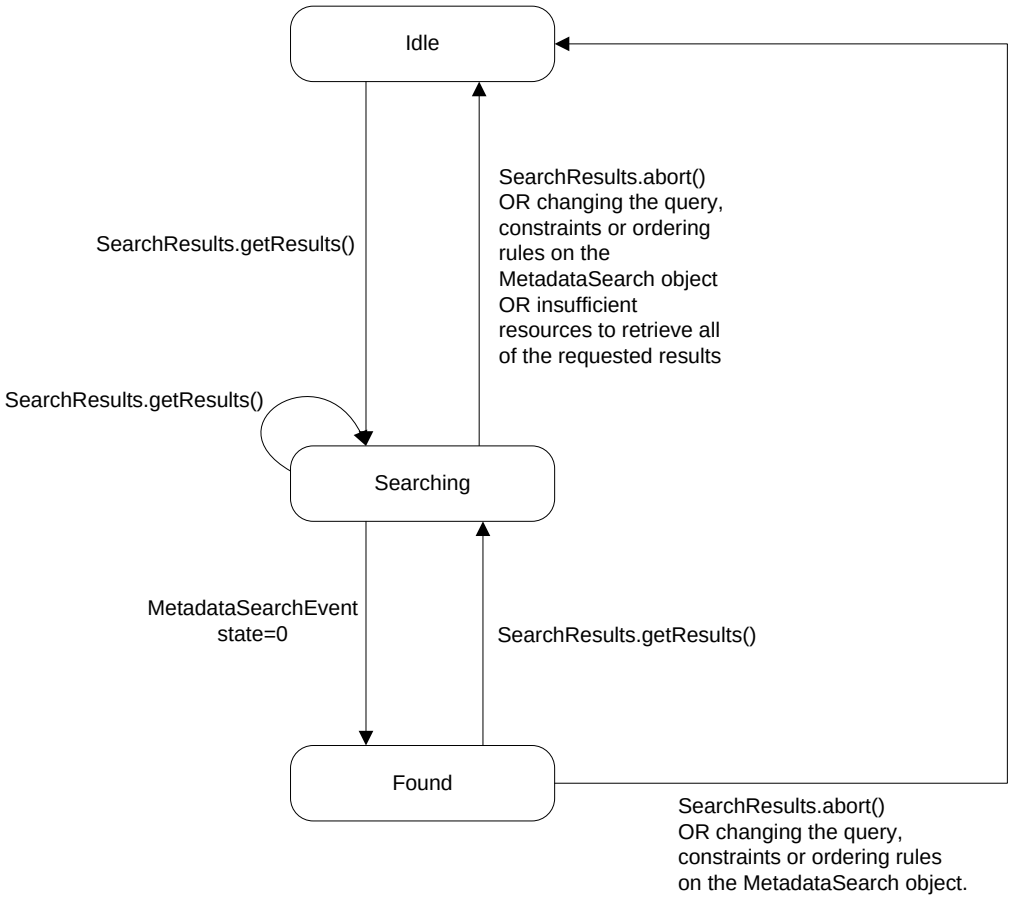

Figure 1 - State machine for a metadata search (informative)

Table A.1 - Metadata search states (normative)

State	Description
Idle	The search is idle; no results are available. This is the initial state of the search. In this state, the application can set or modify the query, constraints or ordering rules that are applied to the search. No search results are available in this state – any calls to `SearchResults.item()` SHALL return `undefined` and the values of the `length` and `totalSize` properties on the SearchResults object SHALL return zero. Any search results that have been cached by the terminal SHALL be discarded when the Idle state is entered. Calling the `SearchResults.getResults()` method SHALL cause a state transition to the Searching state.
Searching	Results are being retrieved and are not yet available to applications. If the terminal has not previously cached the full set of search results, the terminal performs the search to gather the requested results. If a new version of the metadata is detected (e.g. due to an EIT update) while the search is in this state, results SHALL be retrieved from either the new or original version of the metadata but SHALL NOT be retrieved from a combination of the two versions. Calls to `SearchResults.item()` SHALL return undefined. Any modification of the search parameters (e.g. changing the query or adding/removing constraints, or calling `findProgrammesFromStream()`) by the application SHALL stop the current search and cause a transition to the Idle state. The terminal SHALL dispatch a `MetadataSearch` event with `status`=3. When all requested results have been found, the terminal SHALL dispatch a `MetadataSearch` event with `status`=0 and a state transition to the Found state SHALL occur. If the search cannot be completed due to a lack of resources or any other reason, the terminal SHALL dispatch a `MetadataSearch` event with `status`=4 and a state transition to the Idle state SHALL occur.

	Calls to the `SearchResults.getResults()`method SHALL abort the retrieval of search results and attempt to retrieve the newly-requested set of results instead.
Found	Search results are available and can be retrieved by applications. The data exposed via the `SearchResults.item()` method is static and never changes as a result of any updates to the underlying metadata database until `SearchResults.getResults()` is next called. If a new version of the metadata is detected (e.g. due to an EIT update), a `MetadataUpdate` event is dispatched with `action`=1. Subsequent calls to `SearchResult.getResults()` SHALL return results based on the updated metadata. Calls to `SearchResults.getResults()` SHALL cause a state transition to the Searching state. Any modification of the search parameters (e.g. changing the query or adding/removing constraints, or calling `findProgrammesFromStream()`) by the application SHALL cause the current set of results to be discarded and SHALL cause a transition to the Idle state. The terminal SHALL dispatch a `MetadataSearch` event with `status`=3.

Changes to the search parameters (e.g. changing the query or adding/removing constraints or modifying the search target, or calling `findProgrammesFromStream()`)SHALL be applied when the `getResults()` method on the corresponding `SearchResults` object is called. Due to the nature of metadata queries, searches are asynchronous and events are used to notify the application that results are available. `MetadataSearch` events SHALL be targeted at the `application/oipfSearchManager` object.

The present document is intentionally silent about the implementation of the search mechanism and the algorithm for retrieving and caching search results except where described in table 1 above. When performing a search, the receiver MAY gather all search results and cache them (or cache a set of pointers into the full database), or gather only the subset of search results determined by the `getResults()` parameters, or take an alternative approach not described here.

A.2.9.2.2 Properties

`readonly Integer searchTarget`
The target(s) of the search. The value of this property SHALL be 1.

`readonly SearchResults result`
The subset of search results that has been requested by the application.

A.2.9.2.3 Methods

`void setQuery( Query query )`		
Description	Set the query terms to be used for this search, discarding any previously-set query terms.	
Arguments	query	The query terms to be used

`void addChannelConstraint( Channel channel )`		
Description	Constrain the search to only include results from the specified channel. If a channel constraint has already been set, subsequent calls to `addChannelConstraint()` SHALL add the specified channel to the list of channels from which results should be returned.	
Arguments	channel	The channel from which results SHALL be returned. If the value of this argument is `null`, any existing channel constraint SHALL be removed.

`Query createQuery( String field, Integer comparison, String value )`		
Description	Create a metadata query for a specific value in a specific field within the metadata. Simple queries MAY be combined to create more complex queries. Applications SHALL follow the ECMAScript type conversion rules to convert non-string values into their string representation, if necessary.	
Arguments	field	The name of the field to compare. Valid values are “Programme.name”, “Programme.startTime” and “Programme.programmeID”
	comparison	The type of comparison. One of: **Value** | **Description**

		0	True if the specified value is equal to the value of the specified field.
		1	True if the specified value is not equal to the value of the specified field.
		2	True if the value of the specified field is greater than the specified value.
		3	True if the value of the specified field is greater than or equal to the specified value.
		4	True if the value of the specified field is less than the specified value.
		5	True if the value of the specified field is less than or equal to the specified value.
		6	True if the string value of the specified field contains the specified value. This operation SHALL be case insensitive, and SHALL match parts of a word as well as whole words (e.g. a value of "term" will match a field value of "Terminator").
		7	Not included in the present document
	value	The value to check. Applications SHALL follow the ECMAScript type conversion rules to convert non-string values into their string representation, if necessary	

`void findProgrammesFromStream( Channel channel, Integer startTime)`		
Description	Set a query and constraints for retrieving metadata for programmes from a given channel and given start time from metadata contained in the stream as defined in section 4.1.3 of [19]. Searches made using this method will implicitly remove any existing constraints, ordering or queries created by prior calls to methods on this object. This method does not cause the search to be performed; applications must call `getResults()` to retrieve the results.	
Arguments	channel	The channel for which programme information SHALL be found.
	startTime	The start of the time period for which results SHALL be returned measured in seconds since midnight (GMT) on 1/1/1970. The start time is inclusive; any programmes starting at the start time, or which are showing at the start time, will be included in the search results. If `null`, the search SHALL start from the current time.

A.2.9.3 The Query class

A.2.9.3.1 Introduction

The `Query` class represents a metadata query that the user wants to carry out. This may be a simple search, or a complex search involving Boolean logic. Queries are immutable; an operation on a query SHALL return a new `Query` object, allowing applications to continue referring to the original query.

The examples below show how more complex queries can be constructed:

```
Query qa = mySearch.createQuery("Programme.name", 6, "Terminator");
Query qb = mySearch.createQuery ("Programme.programmeID", 0, "dvb://233a.0010.0167");
Query qc = qa.and(qb.not());
```

A.2.9.3.2 Methods

`Query and( Query query )`		
Description	Create a query based on the logical AND of the predicates represented by the current query and the argument query.	
Arguments	query	The second predicate for the AND operation.

`Query or( Query query )`		
Description	Create a query based on the logical OR of the predicates represented by the current query and the argument query.	
Arguments	query	The second predicate for the OR operation.

`Query not()`	
Description	Create a new query that is the logical negation of the current query.

A.2.9.4 The SearchResults class

A.2.9.4.1 Introduction

The `SearchResults` class represents the results of a metadata search. Since the result set may contain a large number of items, applications request a 'window' on to the result set, similar to the functionality provided by the OFFSET and LIMIT clauses in SQL.

Applications MAY request the contents of the result in groups of an arbitrary size, based on an offset from the beginning of the result set. The data SHALL be fetched from the appropriate source, and the application SHALL be notified when the data is available.

The set of results SHALL only be valid if a call to `getResults()` has been made. If this method has not been called, the set of results SHALL be empty (i.e. the value of the `totalSize` property SHALL be 0 and calls to `item()` SHALL return `undefined`).

In addition to the properties and methods defined below a `SearchResults` object SHALL support the array notation to access the results in this collection.

A.2.9.4.2 Properties

`readonly Integer length`
The number of items in the currently available results. The value of this property SHALL be zero until `getResults()` has been called and a `MetadataSearch` event notifying the application that results are available has been dispatched

`readonly Integer offset`
The current offset into the total result set.

`readonly Integer totalSize`
The total number of items in the result set. The value of this property SHALL be undefined until `getResults()` has been called and a `MetadataSearch` event notifying the application that results are available has been dispatched.

A.2.9.4.3 Methods

`Object item( Integer index )`		
Description	Return the item at position index in the collection of currently available results, or `undefined` if no item is present at that position. This function SHALL only return objects that are instances of the `Programme` class.	
Arguments	index	The index into the result set.

`void getResults( Integer offset, Integer count )`		
Description	Perform the search and retrieve the specified subset of the items that match the query. Results SHALL be returned asynchronously. A MetadataSearch event with status=0 SHALL be dispatched when results are available.	
Arguments	offset	The number of items at the start of the result set to be skipped before data is retrieved.
	count	The number of results to retrieve.

`void abort()`	
Description	Abort any outstanding request for results and remove any query, constraints or ordering rules set on the `MetadataSearch` object that is associated with this `SearchResults` object. Items currently in the collection SHALL be removed (i.e. the value of the `length` property SHALL be 0 and any calls to `item()` SHALL return `undefined`). All cached search results SHALL be discarded.

Annex B – Profiles of MPEG DASH

B.1 Introduction (informative)

This annex starts from MPEG DASH [3] and defines a profile that adds additional requirements to improve testability and interoperability.

The present document references only one profile of DASH – the "ISO Base media file format live profile". This profile, on which the HbbTV profile is based, supports both live and on-demand steaming of ISOBMFF content. It supports template-based addressing of short time-aligned Segments that may be concatenated without overlap or video splicing. It supports independently addressable track fragment segments.

B.2 Requirements relating to the MPD

B.2.1 Profile definition

The document defines a sub-profile of the MPEG DASH ISO Base media file format live profile. This sub-profile is identified with the URI "urn:hbbtv:dash:profile:isoff-live:2012" and is called the "HbbTV ISOBMFF Live" profile. All of the requirements and restrictions for the MPEG DASH ISO Base media file format live profile shall apply.

Terminals may raise an error to the application when a referenced MPD does not contain this profile in the `@profiles` attribute. Terminals shall be able to play the content described by the profile-specific MPD (as defined in section 8.1 of DASH [3]) (but not necessarily other Adaptation Sets or Representations in the MPD discarded as part of the process of deriving the profile-specific MPD).

The following clauses of Annex B define the additional restrictions and requirements on an MPD identified as conforming to this profile, as well as requirements on terminals when playing such content.

The size of a MPD shall not exceed 100kbytes.

The content referenced by the profilespecific MPD shall only be encoded using the audio and video codecs defined in section 7.3.1 of TS 102 796 [1].

B.2.2 Numerical requirements

The profile-specific MPD shall conform to the following constraints:

Periods

There shall be no more than "N_{per}" Periods in an MPD that shall be temporally sequential. The behaviour of a terminal is undefined for MPDs containing more than "N_{per}" Periods.

Adaptation Sets

There shall be no more than "N_{adset}" Adaptation Sets per Period in an MPD. The behaviour of a terminal is undefined for MPDs containing Periods with more than "N_{adset}" Adaptation Sets. If there is more than one video Adaptation Set, exactly one must be labelled with a Role@value of "main" from the urn:mpeg:dash:role:2011 CS, to allow the terminal to identify the default adaptation set. Similarly if there is more than one audio Adaptation Set, exactly one must be labelled with a Role@value of "main" to allow the terminal to identify the default adaptation set. There shall be at least one video Adaptation Set per Period in an MPD.

Representations

There shall be no more than "N_{rep}" Representations per Adaptation Set in an MPD. The behaviour of a terminal is undefined for MPDs containing Adaptation Sets with more than "N_{rep} " Representations.

Table B.1 defines these values for the present document:

Table B.1: Maximum numeric requirements on HbbTV ISOBMFF Live MPD

Parameter	Value

N_{per}	32
N_{adset}	16
N_{rep}	16

B.2.3 Metadata Requirements

The profile-specific MPD shall provide the following information for all Representations, whether defined as part of the Representation or inherited.

- For video Representations: @width, @height, @frameRate and @scanType
- For audio Representations: @audioSamplingRate, AudioChannelConfiguration, @lang

Note: @lang is an attribute of the AdaptationSet element and is inherited by its Representations.

B.2.4 Role Related Requirements

The MPD shall adopt the DASH role scheme (urn:mpeg:dash:role:2011) as defined in MPEG-DASH 5.8.5.5, in order that Adaptation Sets can be uniquely differentiated.

Where there are multiple Adaptation Sets of the same component type (e.g. 2 x video Adaptation Sets), terminals shall by default select the Adaptation Set that is signalled with a `Role` element with a value of "main" from the urn:mpeg:dash:role:2011 CS. This does not imply that a terminal must render the "main" Adaptation Set if it understands the logic and signalling of other potentially more appropriate Adaptation Sets or is required by an application to select a different Adaptation Set.

The MPD shall identify audio description streams usingthe `Role` and `Accessibility` descriptors as defined in the following table. Furthermore for receiver mix AD the associated audio stream shall use `depdendencyId` to point out the depdendency to the main representation and hence also point out that the associated audio stream shall not be provided as a representation on it own. Terminals shall ignore audio streams with other `Role` and `Accessibility` descriptor attributes that they do not understand.

Table B.2 : Role and Accessibility descriptor values for Audio Description

		Role descriptor	Accessibility descriptor
schemeIdUri		**urn:mpeg:dash:role:2011**	**urn:tva:metadata:cs:AudioPurposeCS:2007**
value	Broadcast mix AD	alternate	"1"
	Receiver mix AD	commentary	"1"

For example, broadcast mix audio descriptions would be indicated as follows:

```
<Role schemeIdUri=”urn:mpeg:dash:role" value="alternate"/>
<Accessibility schemeIdUri="urn:tva:metadata:cs:AudioPurposeCS:2007" value="1"/>
```

A schematic example for receiver mix audio descriptions:

```
<!-- English Audio, main -->
<AdaptationSet ..>
   <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main" />
   <Representation id="a0" bandwidth="320000"/>
</AdaptationSet>
<!-- English Audio, visually impaired for receiver mixing AD-->
<AdaptationSet ..>
   <Accessibility schemeIdUri="urn:tva:metadata:cs:AudioPurposeCS:2007” value=“1”/>
   <Role schemeIdUri="urn:mpeg:dash:role:2011" value="commentary" />
   <Representation id="a1" dependencyId="a0" bandwidth="64000"/>
</AdaptationSet>
```

B.2.5 Audio Channel Configuration Requirements

In order for the terminals to know the number of audio channels in a representation the MPD should include the Audio Channel Configuration to correctly represent the audio channel configuration

For HE-AAC the Audio Channel Configuration shall use "urn:mpeg:dash:23003:3:audio_channel_configuration:2011" schemeURI with the value set to an integer number as defined in [3]. For example, for a stream with C, L, R, Ls, Rs, LFE, the value shall be "6", as follows:

```
<AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="6"/>
```

For E-AC-3 the Audio Channel Configuration shall use the "urn:dolby:dash:audio_channel_configuration:2011" schemeURI. The value element shall contain a four digit hexadecimal representation of the 16 bit field that describes the channel assignment as defined by Table E.5 in [11] where left channel is MSB. For example, for a stream with L, C, R, Ls, Rs, LFE, the value shall be "F801" (hexadecimal equivalent of the binary value 1111 1000 0000 0001) as follows:

```
<AudioChannelConfiguration
schemeIdUri="urn:dolby:dash:audio_channel_configuration:2011" value="F801"/>
```

B.2.6 Content protection signalling

Content protection signalling is stored within the MPD inside ContentProtection elements (see DASH [3] section 5.8.4.1). The MPD shall contain a ContentProtection element for each content protection system used. MPD URI definitions for ContentProtection elements shall conform to DASH [3] section 5.8.5.2 "Content protection", whereby the method of the third scheme (in the third bullet text) in DASH [3] section 5.8.5.2 shall be applied."

B.3 Restrictions on Content

B.3.1 Restrictions on File Format

B.3.1.1 ISO Base Media File Format

The following restrictions shall apply for content referenced from an profile-specific MPD and carried in the ISO base media file format as defined by ISO/IEC 14496-12 [5]:

- The movie fragment box ('moof') shall contain only one track fragment box ('traf').
- The track run box ('trun') shall allow negative composition offsets (as defined in [9]) in order to maintain audio visual presentation synchronization.

B.3.2 Restrictions on Adaptation Sets

The following additional restrictions shall apply across the set of Representations in an Adaptation Set in a profile-specific MPD:

- Each Representation shall contain only one media component, i.e. a single audio or video track. Other non-media components (e.g. encryption keys) may be present if applicable.
- All ISO BMFF Representations shall have the same track_ID in the track header box and track fragment header box.
- Initialization Segment shall be common for all Representations and the following shall hold:
    - For video Representations, width and height values in track header boxshall have the nominal display size in square pixels after decoding, AVC cropping, and rescaling.
    - All information necessary to decode any Segment chosen from Representations shall be provided in the Initialization Segment. For example, movie box for video Representation shall contain AVC decoder configuration records including all encoding parameters (i.e. Sequence Parameter Sets ans Picture Parameter Sets) used for Representations in the Adaptation Sets.

NOTE: Common initialization segment means that all representations in an adaption set have identically the same 'stsd' box. There will be one entry in the 'stsd' box for each representation. Representations encoded with different "parameters" will use the sample_description_index in the Track Fragment Header to identify which of the sample entries in the 'stsd' box is applicable to them.

- Each Segment shall consists of a whole, self-contained movie fragment.
- Segments shall be at least 1s long, except for the last segment in an MPD which may be shorter.
- Each video Segment shall have a duration of not more than fifteen seconds.
- Each audio Segment shall have a duration of not more than fifteen seconds.

There is no requirement for all of the transitions between all the Representations of a media content component to be ones that terminals are required to support as defined in clause B.4.2 Transitions between Representations. Adaptation Sets may include Representations which can only be reached by transitions other than those which terminals are required to support.

B.4 Requirements on Terminals

B.4.1 DASH Profile Support

Terminals shall support the HbbTV ISOBMFF Live profile. Other profiles may be supported.

B.4.2 Transitions between Representations

B.4.2.1 Video Tracks

During playback of adaptively streamed content encoded using AVC, terminals shall support transitions between video Representations as follows:

1) Between Representations which differ by bit-rate (note 1).

2) Between Representations which differ by profile and/or level (note 2).

3) Between Representations which differ by full-screen resolution (e.g. 1920x1080 and 720x576) (note 2) (note 3).

4) Between Representations with the same full-screen resolution but different luminance resolutions as defined in table 9 “Table 9: Resolutions for Full-screen Display from 25 Hz H.264/AVC SDTV IRD and supported by 25 Hz H.264/AVC HDTV IRD, 50 Hz H.264/AVC HDTV IRD, 25 Hz SVC HDTV IRD and 50 Hz SVC HDTV IRD” and table 12 “Resolutions for Full-screen Display from H.264/AVC HDTV IRD and SVC HDTV IRD” of TS 101 154 [8] (e.g. 1920x1080 and 1440x1080) (note 2).

NOTE 1: Transitions shall be seamless unless combined with other changes which do not have that requirement.

NOTE 2: Transitions may include repeated frames but shall otherwise be seamless.

NOTE 3: As defined in clause 10.2.1 of HbbTV [1], video shall be scaled, preserving the aspect ratio, such that all of the decoded video is visible within the area of the AV Control object. Clause 5.5.3.1 of MPEG DASH [3] requires all Representations in an AdaptationSet to have the same picture aspect ratio. The resolution and pixel aspect ratio can change as long as the picture aspect ratio remains the same.

Some examples of transitions between Representations which terminals may support but which are not required to support include:

1) Between Representations where one is interlaced and the other is progressive

2) Between Representations which differ in framerate, e.g. 25 and 50 fps

Terminals should not make transitions between Representations that would cause noticeable disruption to the presentation of the media at the switch point unless the transition is necessary to prevent interruption to the media presentation due to lack of data.

B.4.2.2 Audio tracks

During playback of adaptively streamed content encoded using HE-AAC or E-AC3, terminals shall support transitions between audio Representations as follows:

1) Between Representations which differ by bit-rate (note 1).

NOTE 1: Transitions shall be seamless unless combined with other changes which do not have that requirement.

Some examples of transitions between Representations which terminals may support but which are not required to support include:

1) Between Representations where one is encoded with HE-AAC and the other is E-AC3.

2) Between Representations which differ in the number of audio channels.

3) Between Representations which differ in the sampling frequency.

B.4.3 Buffering

The terminal should not buffer more than data equivalent to approximately 300 seconds of normal play in advance of the current play position.

The requirement in clause 10.2.3.2 of HbbTV [1] concerning persistent storage of streamed content shall also apply to content delivered as specified in this annex.

B.4.4 ISO File Format Support

Terminals shall support more than one sample entry in the 'stsd' box and shall support the use of the sample_description_index in the Track Fragment Header at the start of each segment to identify which of the sample entries is applicable to that segment.

Annex C: Common Encryption for ISOBMFF

Support for MPEG common encryption as defined in CENC [4] is optional in the present document. If it is supported then the following requirements shall apply.

C.1 Key Management for On Demand Content

The HbbTV ISOBMFF Live media files shall be encrypted using a single key for all Representations and all media components in all Periods, and a single KID. As a consequence, the same key is used for all Representations of an On Demand asset, independent of its duration.

NOTE: In cases where it is desired to use different keys for different Representations or media components, this may be done using multiple MPDs. For example, in order to target multiple groups of users or multiple device classes.

C.2 Key Management for Live Content

The HbbTV ISOBMFF Live media files shall be encrypted using a single key for all Representations and all media components within a single Period.

NOTE Periods are typically used for separate programs in a live broadcast.

NOTE: In cases where it is desired to use different keys for different Representations or media components, this may be done using multiple MPDs. For example, in order to target multiple groups of users or multiple device classes.

The KID may be updated but not faster than every 120 seconds.

As a consequence, while the same key is used for all Representations of alive asset, the key may be updated on a regular basis, hence reproducing a lower frequency key update mechanism than the one usually used to protect broadcast signals.

C.3 Encryption mode

Media data shall be encrypted using AES 128-bit in CTR mode (AES-CTR) as defined in section 9 of CENC [4].

C.4 Usage of ISOBMFF boxes

This clause specifies relevant parameters of existing ISOBMFF boxes used with CENC [4].

C.4.1 ‘pssh’ box

An ISOBMFF file may contain multiple Protection System Specific Header ('pssh') boxes (as defined in CENC [4]). The terminal shall be able to identify and use the 'pssh' box that corresponds to the DRM system that is available to the terminal. If the terminal has multiple DRM systems available with matching 'pssh' boxes, the terminal shall select between them to decrypt the content.

Usage of the ‘pssh’ by the DRM in either ‘moov’ or ‘moof’ box is optional. Normally, information in the MPD is sufficient for license acquisition by the terminal, but in live streaming situations, it may be necessary to distribute new protected keys/licenses in a ‘pssh’ box in each downloaded Track Fragment to allow encryption changes during a presentation (i.e. “key rotation”, multiple programs, interspersed advertisements, etc.).

If a DRM system uses the 'pssh' box, then the value of the SystemID field corresponding to that DRM system shall be specified as well as the encoding of the Data field.

C.5 Extensions to ISOBMFF boxes

C.5.1 Constraints on the SampleAuxiliaryInformationOffsetsBox

In order to ensure that the terminal has access to the sample auxiliary information before it is needed to decrypt a sample, the offsets in any 'saio' box must be such that they point to data that is located before the sample media data to which this sample auxiliary information corresponds.

For example, each ‘traf’ box of a track that may contain encrypted media samples may contain a Sample Encryption Information box (‘senc’) to provide the initialization vectors and subsample encryption information necessary to decrypt any encrypted media samples using the CENC [4] as defined in Section 7 of that document.

Box Type ‘senc’

Container Track Fragment Box (‘traf’)

Mandatory No

Quantity Zero or one

Syntax

```
aligned(8) class SampleEncryptionBox extends FullBox('senc', version=0, flags=0) {
      unsigned int(32)  sample_count;
      {
            unsigned int(IV_size*8)  InitializationVector;
            if (flags & 0x000002)
            {
                  unsigned int(16)  subsample_count;
                  {
                        unsigned int(16)  BytesOfClearData;
                        unsigned int(32)  BytesOfEncryptedData;
                  } [ subsample_count ]
            }
```

```
      }[ sample_count ]
}
```

Annex D – DRM Integration (informative)

D.1 Introduction

This annex identifies issues which need to be considered and in most cases documented when defining how a DRM system is to be integrated with HbbTV. It is expected that solutions to these issues would form the basis of the document defining the technical integration between HbbTV and that DRM system and subsequently a test specification and test suite.

D.2 General issues

Some informative text is needed identifying how the key aspects of the DRM technology map on to the mechanisms and local interfaces showin in annex D of OIPF volume 5 [6].

A DRM System ID for the DRM system needs to be registered in as described in OIPF Volume 5 [6], Section 9.3.10.

If the DRM agent can generate user interfaces on the terminal then the interaction between these and the HbbTV system needs to be defined. This is particularly critical if these user interfaces are rendered using the same browser as is used for HbbTV applications. (See OIPF Volume 5 [6], section 5.1.1.6).

Which combinations of protocols and codecs are required to be supported with the DRM technology need to be defined. These must be in the format of the video profile capability strings indicating as defined in OIPF Volume 5 [6], section 9.3.11.

D.3 DRM Agent API

In the `sendDRMMessage` method (as defined in OIPF volume 5 [6], section 7.6.1.2), it needs to be defined which values of the `msgType` parameter are valid and what the contents of the `msg` parameter are for each message type.

In the `onDRMMessageResult` function (as defined in OIPF Volume 5 [6], section 7.6.1.1), the valid values for the `resultMsg` parameter should be defined if they are intended to be parsed by an HbbTV application. Additionally it needs to be defined which conditions in the DRM system trigger which `resultCode` values and any implications on the value of the `resultMsg`.

D.4 Content via the CEA-2014 A/V Object

If DRM is used to protect content presented via the CEA-2014 A/V object then the following need to be specified;

1) Whether the content access streaming descriptor is needed to provide information for the DRM system. If so then which of the fields are used, under what circumstances and what the requirements are on their contents need to be defined. If not then the mechanism by which DRM information is obtained needs to be defined.

2) Whether the DRM system can enforce parental access control and trigger an `onParentalRatingChange` event (as defined in OIPF volume 5 [6], section 7.14.5). If this event can be triggered then how the value of the `contentID parameter` is obtained needs to be specified. The same applies for `onParentalRatingError` event.

3) The conditions when the `onDRMRightsError` event is generated (as defined in OIPF Volume 5 [6], section 7.14.6). If it is generated, the values to be used for the `contentID` and the `rightsIssuerURL` parameters need to be defined.

Annex E – Code Examples (informative)

E.1 Access to EIT Schedule Information

```
////////////////////////////////////////////////
//
//  Event listeners and functions to process the search results.
//  These functions are common to both examples
//
////////////////////////////////////////////////


var mySearch;

// Listener for events relating to a metadata search
function searchUpdate(state, id) {

    if (state == 0) {
        // the search is complete

        // update the search results to make sure we have all results
        mySearch.result.update();

        processSearchResults();
    }
    else if (state == 1) {
        // more results are available

        // ignore this for now and wait for all data to be available
        // we could use this to incrementally update our UI if we wanted
    }
}


// Process the search results when the search is completed
function processSearchResults() {

    var results = mySearch.result;

    // Iterate over the results and do stuff
    var i;
    for (i=0; i < results.length; i++) {
        var name = results[i].title;
        var startTime = results[i].startTime;
        var duration = results[i].duration;

        // do stuff
    }
}



////////////////////////////////////////////////
//
//  Example 1- Get programme schedule data for a set of channels
//
////////////////////////////////////////////////

function getGuideData() {

    // Instantiate the search manager
    var mySearchMgr = window.oipfObjectFactory.createSearchManagerObject();
    mySearchMgr.addEventListener("MetadataSearch", searchUpdate);

    // Create the search object to search EPG data
    mySearch = mySearchMgr.createSearch(1);

    // Get the start time for the search.  We round down to the previous
// half-hour boundary
    var now = Math.round(new Date().getTime()/1000.0);
    var gridStart = (Math.floor(now/1800)) *1800)

    //Now use that to set the start and end times for the search
    var startQuery = mySearch.createQuery("startTime", 3, gridStart);
```

```
    // Assume the grid holds two hours worth of programmes
    var endQuery = mySearch.createQuery("startTime", 4, gridStart + 7200);

    // Now set that query
    mySearch.query = startQuery.and(endQuery);



    // Start the search, getting the first 50 results
    if (mySearch.result.getResults(0, 50)) {
        // All our data is available immediately
        processSearchResults();
    }
    else {
        // We need to wait...
    }
}



////////////////////////////////////////////////
//
//  Example 2 - Get now/next data for a channel other
//  than the currently selected one.
//
////////////////////////////////////////////////

function getNowNextData() {

    // Instantiate the search manager
    var mySearchMgr = window.oipfObjectFactory.createSearchManagerObject();
    mySearchMgr.addEventListener("MetadataSearch", searchUpdate);

    // Create the search object to search EPG data
    mySearch = mySearchMgr.createSearch(1);

    // Get the current time
    var now = Math.round(new Date().getTime()/1000.0);


    // Find data for the first channel in the channel list
    var chan = mySearchMgr.getChannelConfig().channelList[0];

    // Start the search, getting the first 2 results
    if (mySearch.findProgrammesFromStream(chan, now, 2)) {

        // All our data is available immediately, so we don't need to call
// update() on the search results.
        processSearchResults();
    }
    else {
        // We need to wait...
    }
}
```