Marlin Dynamic Media Zones

Version 1.1
Final

Source	Marlin Developer Community
Date	January 26, 2011

Refer to Notices on page 2 for important legal information

Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, CONCERNING THE COMPLETENESS, ACCURACY, OR APPLICABILITY OF ANY INFORMATION CONTAINED IN THIS DOCUMENT. THE MARLIN DEVELOPER COMMUNITY ("MDC") ON BEHALF OF ITSELF AND ITS PARTICIPANTS (COLLECTIVELY, THE "PARTIES") DISCLAIM ALL LIABILITY OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, ARISING OR RESULTING FROM THE RELIANCE OR USE BY ANY PARTY OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. THE PARTIES COLLECTIVELY AND INDIVIDUALLY MAKE NO REPRESENTATIONS CONCERNING THE APPLICABILITY OF ANY PATENT, COPYRIGHT (OTHER THAN THE COPYRIGHT TO THE DOCUMENT DESCRIBED BELOW) OR OTHER PROPRIETARY RIGHT OF THIS DOCUMENT OR ITS USE, AND THE RECEIPT OR ANY USE OF THIS DOCUMENT OR ITS CONTENTS DOES NOT IN ANY WAY CREATE BY IMPLICATION, ESTOPPEL OR OTHERWISE, ANY LICENSE OR RIGHT TO OR UNDER ANY PATENT, COPYRIGHT, TRADEMARK OR TRADE SECRET RIGHTS WHICH ARE OR MAY BE ASSOCIATED WITH THE IDEAS, TECHNIQUES, CONCEPTS OR EXPRESSIONS CONTAINED HEREIN.

Use of this document is subject to the agreement executed between you and the Parties, if any.

Any copyright notices shall not be removed, varied, or denigrated in any manner.

Intellectual Property

A commercial implementation of this specification requires a license from the Marlin Trust Management Organization.

Contact Information

Feedback on this specification should be addressed to: editor@marlin-community.com

Contact information for the Marlin Trust Management Organization can be found at: http://www.marlin-trust.com/

Contents

1 Introduction

This specification defines an extension to the Marlin Core specification that provides support for the description of different types of zones in media presentations. Media zones are portions of media content that have specific attributes representing constraints that a media player application must obey when playing back the content, such as advertisement zones that must not be skipped, or a warning screen that must be viewed before the rest of the presentation can be viewed.

Note that this specification may be used with unencrypted content, In this case, §4, 5, and 6 and any requirements on signatures do not apply.

1.1 *Terminology and Conventions*

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this specification are to be interpreted as described in IETF [RFC2119].

1.2 *References*

1.2.1 Normative References

[DVB SI]	ETSI EN 300 468 V1.11.1 (2010-04) Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems
[IEC62455]	Internet protocol (IP) and transport stream (TS), based service access IEC 62455 © IEC:2007(E) First edition 2007-06
[MarlinBB]	Marlin Broadband Delivery System Specification, Version 1.2
[RFC2104]	HMAC: Keyed-Hashing for Message Authentication, IETF RFC 2104 http://www.ietf.org/rfc/rfc2104.txt.
[RFC2119]	S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.
[RFC3174]	US Secure Hash Algorithm 1 (SHA-1), IETF RFC 3174 http://www.ietf.org/rfc/rfc3174.txt.
[13818-1]	ISO/IEC 13818-1:2000. Information technology — Generic coding of moving pictures and associated audio information: Systems
[14496-12]	ISO/IEC 14496-12:2003. Information technology -- Coding of audio-visual objects -- Part 12: ISO Base Media File Format
[14496-1]	ISO/IEC 14496-1:2004 subpart 3. Information technology — Coding of audio-visual objects

1.3 *Terms and Definitions*

uimsbf	unsigned integer, most significant bit first

121

2 Media Zones

The complete definition of media zones requires precise positioning information within media presentations, thus requiring some media-format specific elements. This specification defines abstract data structures and elements common to all media formats. In addition, specific ways of representing and embedding these data structures and elements to ISO Base Media File Format [14496-1] and MPEG2 TS File Format [13818-1] are defined in §7. Mapping of the abstract data types onto other media formats may be defined in updates to this specification or in other specifications.

2.1 *Zone Map*

A zone map is a table that contains zone descriptions. Zones are spans of media streams. A span of media stream is bounded by two points in that stream. Each point record contains a reference to a random access point in a media stream (such as a sample, or access unit, in a media format derived from the ISO Base Media Format). An internal zone is defined by specifying a start and end point in the media stream with which it is associated. An external zone is defined by specifying two things: (1) a point in the current file's media stream at which a portion of another media stream (from a different file) is to be spliced, and (2) an ID for that external zone. The start and end points of the external zone's media are specified in an internal zone defined in the zone map for the file containing that media.

A zone map also has a signature that prevents the zone descriptions from being modified by entities that do not have knowledge of a specific key, whose value is described in the table beneath the signatureAlgorithm description below.
The abstract data types that make up a zone map are:

```
ZonePoint: {
      accessUnitReference: <media format dependent>
}
InternalZoneInfo: {
      fromPoint: integer
      toPoint: integer
      id: integer
      attributes: integer
      mediaDigestAlgorithm: integer
      mediaDigestValue: byte array
      meteringTag: string
}
ExternalZoneInfo: {
      splicePoint: integer
      id: integer
}
Extension: {
      type: integer
      criticalFlag: bit(8)
      payload: byte array
}
ZoneMap: {
      points: array of ZonePoint
      internalZones: array of InternalZoneInfo
```

```
      externalZones: array of ExternalZoneInfo
      extensions: array of Extension
      signature: {
            signatureAlgorithm: integer
            signatureValue: byte array
      }
}
```

`accessUnitReference`: reference to, or identifier of, a position or access unit of the media.

`fromPoint`: index of a `ZonePoint` element of the points array where the zone media starts.

`toPoint`: index of a `ZonePoint` element of the points array where the zone media ends. The media considered to be in the zone is the media up to, but not including, `toPoint`. That is, the zone media is considered to start at `fromPoint` (inclusive) and end at `toPoint` (noninclusive).

`splicePoint`: index of a `ZonePoint` element of the points array where the zone media is to be spliced. If a zone is spliced at point B, that zone media is played before the media starting at point B.

`id`: identifier for the zone.

`attributes`: bit vector equal to a combination of zero or more of the following flags:

Attribute	Value	Description
INSERTED	1	This zone represents a portion of the presentation that has been inserted in the main presentation (such as an advertisement) and MAY be skipped by the player unless the zone has a NOSKIP obligation.
Marlin Reserved	2	Reserved for later use by MDC
Marlin Reserved	4	Reserved for later use by MDC
NOSKIP	8	Corresponds to the Zone Type identified with NOSKIP in §3.
MAGNETIC	16	Corresponds to the Zone Type identified with MAGNETIC in §3.
STICKY	32	Corresponds to the Zone Type identified with STICKY in §3.
Flag BIT 1	64	Flag BIT 1 and Flag BIT 2 together may be used to assign a 4-level priority to the zone. Flag BIT 2 is the most significant bit.
Flag BIT 2	128	

When a Marlin License is present for this content the INSERTED attribute MUST be observed.

When a Marlin License exits for the content, the Marlin DRM Client MUST ignore NOSKIP, MAGNETIC, STICKY, Flag Bit 1 and/or Flag Bit 2 in `attributes`.

Note: this specification does not define normative behavior associated with the Flag Bits. Such behavior may be defined in specifications referencing this specification.

`mediaDigestAlgorithm:` identifier of the digest algorithm used to compute the `mediaDigestValue` field. The following algorithm identifiers are defined:

Name	Value	Description
NONE	0	The mediaDigestValue bytes array MUST be empty
SHA1	1	The bytes of the mediaDigestValue are obtained by computing the SHA-1 [RFC3174] hash of the zone's media byte stream.

`mediaDigestValue:`: media-dependent digest of the media samples that are part of the zone. For each supported media type, a media-dependent specification will specify a media byte stream used as input to a hash function.

`meteringTag:` string used as a tag for reporting zone playback metering (see §6).

`points`: array of one or more `ZonePoint`.

`internalZones`: array of zero or more `InternalZoneInfo`.

`externalZones`: array of zero or more `ExternalZoneInfo`.

`extensions`: array of zero or more `Extension`

`type`: by convention, the 32-bit identifier for an `Extension` is written as a 4-letter word, where each letter's 8-bit ASCII code is the corresponding 8-bit byte portion of the identifier. For example, the identifier value 0x61626364 (hexadecimal) would be written 'abcd', because the ASCII code for 'a' is 0x61, etc.

`criticalFlag`: bit vector of flags. An `Extension` that is marked Marlin critical (by the bit 0(LSB) of `criticalFlag` set to 1) SHALL be enforced. If an `Extension` marked as critical is encountered that is not supported or understood, then the content SHALL NOT be rendered. Extensions that are not supported or understood and that are not marked critical SHOULD be ignored. A Marlin DRM Client SHALL ignore all other bits of this vector. Note that this allows marking extensions critical if this specification is used outside the context of Marlin.

`payload`: description of `Extension`.

Note that `Extensions` may be used to add functionality. This version of the specification does not define any `Extensions.`

`signature`: keyed-MAC signature of the `points` and `internalZones` and `externalZones` and `extensions` arrays. The MAC algorithm and key is specified by the `signatureAlgorithm` field.

`signatureAlgorithm`: identifier of the signature algorithm used to compute the signatureValue field. The following algorithm identifiers are defined:

Name	Value	Description
HMAC_SHA1_HMK	0	The signature value is obtained using HMAC as defined in [RFC2104] using the SHA-1 hash function as defined in [RFC3174]. The length of the signature key MUST be 16 bytes. Unless there is a media format specific rule to select the signature key, the signature key is obtained by taking the first 16 bytes of the SHA-1 hash of the byte array consisting of the 4 constant bytes 0x4d, 0x44, 0x4d, and 0x5a followed by the bytes of the ZoneMapKey as defined in §2.3

`signatureValue`: value of the signature as specified by the `signatureAlgorithm` field. When there is no applicable signature key for a certain media (e.g. clear-text content), the signature value is omitted by setting a signature key length to 0. The signature key length corresponds to 'signatureValueDataSize' for ISO Base Media File Format as in Appendix A, and it corresponds to 'signature_value_length' for MPEG2-TS File Format as in Appendix C.

2.1.1 External Zones

External zones referenced in a zone map are zones for which the associated media is not in the same file or container as the media associated with that zone map. External zones allow the media for a zone to be delivered or packaged separately from the main media presentation in which they will be rendered. When a zone map contains references to external zones, the media with the content for those zones MUST have an associated zone map with internal zone descriptions for them. When more than one zone is spliced at the same point, the playback order is the order in which the zones appear in the `externalZones` array.

This document specifies the OPTIONAL mechanism (zone location) by which an application can locate the media and associated zone maps for external zones in §2.4. When the zone location is not specified, these issues are out of the scope of this specification.

2.1.2 Zone Identifiers

Zone identifiers are integers used to identify each zone. Zone identifiers allow an application to locate the appropriate media zone when signaled in a MediaZones obligation (as defined in §3). Identifiers are local to a specific content. It is the ZoneMapKey (defined in §2.3) used to sign the zone map that ties zones to a specific content.

There MAY be more than one zone with the same identifier in a zone map, and there MAY be more than one zone map containing zones with the same zone identifier. In this case, any valid zone can be used. The possibility of having more than one zone with the same identifier allows for models where multiple different media streams for a zone are delivered to a player application that then chooses one of the valid zones when rendering a media presentation.

This document does not specify the mechanism by which an application chooses between multiple zones with the same identifier.

Note: zone identifiers may be used as “behavioral identifiers” to indicate obligations such as NOSKIP, MAGNETIC or STICKY as explained in Appendix E.6. This usage of zone identifier allows defining the same behavior for more than one zone in the same media stream.

2.2 *Zone Digests*

When the media player plays a zone that has a `digestAlgorithm` different from `NONE (0)`, it MUST compute the zone digest as it is being played. When the zone has been completely played, it MUST compare the value of the computed digest with the value of the `mediaDigestValue` field for that zone. If the values are not equal, the zone is not intact: the player application MUST stop any further playback of the presentation, and MUST NOT call any OnZoneCompleted callback as described in §5. It also MUST NOT log a metering event, as described in §6.

2.3 *Zone Map Key*

The key that is used to bind the zone map to a specific media content is derived from one of the content keys used to encrypt the content. When the control for a license returns an ESB with a MediaZones obligation (§3), the zone map signature MUST be verified prior to using the zone map information. In order to verify the signature, the signature key must be obtained. As described in §2.1, part of the signature key calculation may utilizes the ZoneMapKey. Unless there is a media format specific rule to select the signature key and when there is more than one content key (for example when playing media from a container where the audio and video streams are encrypted with different content keys), the ZoneMapKey is selected as follows:

For audio-only media, the ZoneMapKey is the content key used to encrypt the audio stream.

For video-only or audio+video media, the ZoneMapKey is the content key used to encrypt the video stream.

2.4 *Zone Location*

A zone location is a table that MAY contain one or more set of zone resource location and its access scheme for an external zone identified with `id` in a zone map and zone location. When a zone map includes an external zone, a corresponding zone location MAY be included for the external zone to explicitly indicate one or more acquisition method of an external media corresponding to the external zone.

The abstract data types that make up a zone location are:

```
SchemeLocationInfo: {
      scheme: unsigned integer (32)
      locationValue: byte array
}
ZoneLocationInfo: {
      id: integer
      schemeLocationElements: array of SchemeLocationInfo
}
```

```
ZoneLocationMap: {
      zoneLocationElements: array of ZoneLocationInfo
}
```

`scheme`: method to be used to access `locationValue`. (e.g. http). This is a 32-bit identifier in big-endian byte order. By convention, the 32-bit identifier for a scheme is written as a 4-letter word, where each letter's 8-bit ASCII code is the corresponding 8-bit byte portion of the identifier. For example, the identifier value 0x68747470 (hexadecimal) would be written 'http', because the ASCII code for 'h' is 0x68, etc…

`locationValue`: resource location value to be accessed by the method identified by `scheme`. (e.g. http://foo.bar)

`id`: identifier for the zone.

When an external zone identified by a certain `id` is included in a zone map, a corresponding zone location identified with the same `id` MAY provide a location of external media corresponding to the external zone. The zone location MAY include one or more acquisition methods identified by the `scheme`, so a player application SHOULD choose its supported `scheme` and its `locationValue` to access to the external media.

3 Zone Types and Behavior

This section specifies the following Zone Types identified with names and the corresponding mandatory behavior of the player:

Name	Description
NOSKIP	The player application MUST NOT automatically skip this zone: the zone MUST be played as an integral part of the presentation. This type does not, however, prevent fast-forwarding or skipping once zone play has begun.
MAGNETIC	This zone is magnetic: if the player application attempts to seek to a position inside the zone, then the playback MUST begin at the 'fromPoint' point of the zone.
STICKY	This zone is sticky: if the player enters this zone, it MUST disable the ability to fast-forward or to skip this zone until the playback position is outside the zone.

It is possible that the zones described in the MediaZones list overlap. Also, the same zone may be included more than once with different zone types, in which case that zone has the combined properties of all those types.

When the media player application attempts to seek to a position that is inside more than one MAGNETIC zone, the playback MUST begin at the earliest 'fromPoint' of all those zones.

Note: When zone identifiers are used to imply behaviors (NOSKIP, MAGNETIC or STICKY), obligations will be enforced for each zone as defined in Zone Map with predefined behavioral zone identifiers in a Control of a license.

4 MediaZones Obligation

A Marlin Control MAY take advantage of the support for Zones by including a MediaZones obligation in the ESB returned by the 'Check' and/or 'Perform' methods of a playback-related action (such as 'Play').

The following constraint MAY be included in the Obligations container of an ESB:

Name	Type	Description
MediaZones	ValueList	One or more ZoneInfo records. Each ZoneInfo record is a ValueList with the following values: Type: Integer — Zone Id equal to the 'id' field of one of the zones in the media's zone map. Type: Integer — Zone type identifier. The possible type values are defined below. Type: Integer — Bit-vector of zero or more OR'ed flag values. The possible flag values are defined below.

The following ZoneInfo flag values are defined:

Name	Value	Description
METER	1	If there is a metering obligation for this content, the application MUST also log a metering event when this zone has been played. (see §6)
INCLUDE_SPLICE	2	If this flag is set the span of media inside this zone includes the media in the zones spliced at its 'end' point, if any. If this flag is not set, the span of media inside this zone does not include the zone(s) spliced at its 'end' point.

Zone Type Identifiers

Name	Identifier	Description
NOSKIP	0	Corresponds to the Zone Type identified with NOSKIP in §3.
MAGNETIC	1	Corresponds to the Zone Type identified with MAGNETIC in §3.
STICKY	2	Corresponds to the Zone Type identified with STICKY in §3.

To comply with a MediaZones obligation, a player application MUST locate at least one valid `InternalZoneInfo` entry, either in the media file itself, or in an external media presentation to be spliced in as an external zone, for each of the zone identifiers specified in the ZoneInfo records of the obligation. A valid entry is an entry contained in a zone map for which the signature is valid.

5 Zone Callbacks

A Marlin Control may include an OnZoneCompleted callback notice in the ESB returned by the 'Check' and/or 'Perform' methods of a playback-related action (such as 'Play').

Name	Type	Description
OnZoneCompleted	ValueList	The Host application MUST callback when the specified zone has been completely played (unless the zone is found not to be intact as specified in §2.2) The values in the ValueList are: Type: Integer — Zone Id equal to the 'id' field of one of the zones in the media's Zone Map Type: Callback — Routine to call back, and associated cookie.

6 Zone Metering

When playing content with a CRITICAL metering obligation as defined in §6.3 of [MarlinBB], if a zone has completely been played and the METERING flag is set for that zone in the MediaZones obligation, the player application MUST log that event (unless the zone is found not to be intact, as specified in §2.2). When reporting that event in metering data to the metering service, the entry corresponding to that event SHALL be an Event record as defined in §6.3.2 of [MarlinBB]. The record SHOULD only include the "stop" time, omitting the "start" time.

The logical id for the metering record SHALL be the string obtained by concatenating the logical id of the metering obligation, the character '#', and the `meteringTag` field for that zone.

If the content does not have a metering obligation, a player application MUST ignore any METERING flag. If the content has a metering obligation that is not marked as CRITICAL, the player application MAY ignore the METERING flag.

7 Mapping of MediaZones

This section describes the mapping of DMZ zone maps and zone locations to ISO Base and MPEG2-TS file formats.

7.1 *ISO Base Media File Format Mapping*

The definitions of boxes below are given in the syntax description language (SDL) defined in [14496-1].

7.1.1 Zone Map

When defining zone points for media content derived from the ISO Base Media File Format [14496-12], the `accessUnitReference` field of the ZonePoint structure SHALL be an IsoMediaAccessUnit for sample-based formats, or a IsoMediaByteOffset for box-based formats (such as OMA DCF).

```
IsoMediaAccessUnit: {
    sample: integer
}
```

`sample`: an integer equal to a 1-based sample number for the media track to which the zone corresponds.

```
IsoMediaByteOffset: {
    offset: integer
}
```

`offset`: an integer equal to a 0-based offset from the beginning of the cleartext data for this media.

For backward compatibility with Version 1.0 of this specification, the zone map associated with a track SHOULD be included as an ‘mZON’ box in a ‘udta’ container box in the ‘trak’ box for that track (track-based media) or the ‘udta’ container box in the Discrete Media headers box (OMA DCF media). When supporting the new extensible syntax, the zone map associated with a track SHOULD be included as an ‘mDMZ’ box. Unless backward compatibility is necessary, it is RECOMMENDED to use the new extensible syntax.

```
class MediaZoneMap() extends Box (‘mZON‘) {
     uint(32) zoneMapDataSize;
     bit(8)   zoneMapData[zoneMapDataSize];
}
```

`zoneMapDataSize`: size of the binary encoding representation of the zone map.
`zoneMapData`: binary encoding of the zone map (Appendix A).

For track-based audio-only presentations, the zone map SHALL be included in the audio track.

For track-based audio+video presentations, the zone map SHALL be included in the video track.

For box-based presentations, the zone map SHALL be a descendent of the same container as the one that contains the media data box for the presentation to which the map corresponds.

For media where the media data consists of access unit samples, the `mediaDigestValue` for a zone is computed over the byte sequence made of all the sample data beginning with the first sample (the media at the zone fromPoint) and ending with the last sample of the zone (the sample immediately preceding the toPoint).

For media where the media data is represented by a single byte sequence (such as the 'odda' box in OMA DCF), the `mediaDigestValue` for a zone is computed over the portion of the byte sequence beginning with the first byte of the zone (at fromPoint) and ending with the byte immediately preceding that specified by toPoint.

7.1.2 Zone Location

The zone location associated with a zone map SHOULD be included as an 'zloc' box in a 'udta' container box in the 'trak' box for that track (track-based media) or the 'udta' container box in the Discrete Media headers box (OMA DCF media). When the zone location is included, the zone location SHALL be included in the same 'trak' where the corresponding zone map is included.

```
class ZoneLocationMap() extends Box ('zloc') {
     uint(32) zoneLocationMapSize;
     bit(8)   zoneLocationMap [zoneLocationMapSize];
}
```

`zoneLocationMapSize`: size of the binary encoding representation of the zone location.

`zoneLocationMap`: binary encoding of the zone location (Appendix B).

7.2 *MPEG2-TS File Format Mapping*

MPEG2-TS is used to build a single digital transport stream or multiplex, which can carry one or more programs. MPEG-2 TS is defined in [13818-1]. Programs/Services using the mechanism defined in this document SHALL include, in the program info descriptor loop of the PMT, a private_data_specifier_descriptor, as defined in [DVB SI] with the private_data_specifier set to [value to be obtained from DVB through registration, http://www.dvbservices.com/identifiers/] and a DMZ descriptor as defined in §7.2.1. Furthermore, these Programs/Services SHALL contain zone map information as private data according to §7.2.3 and MAY contain zone location information as private data according to §7.2.4.

Note that this specification defines a media format-specific rule for MPEG2-TS, for selecting the signature key. That is, a dedicated signature key is transmitted inside the DMZ Access Criteria Descriptor in ECMs.

7.2.1 DMZ Descriptor in Program Map Table

The DMZ descriptors SHALL be formatted as specified in **Table 1**.

Table 1 – DMZ descriptor

Syntax	No. of bits	Mnemonic	Value
DMZ descriptor() {			
descriptor_tag	8	uimsbf	128
descriptor_length	8	uimsbf	2
Reserved	3	bslbf	
PID	13	uimsbf	
}			

descriptor_length, the length of the descriptor

PID, the PID on which the zone maps and possibly zone locations defined in §7.2.3 and §7.2.4 can be found.

7.2.2 DMZ Access Criteria Descriptor in ECM

This section defines an access_criteria_descriptor for DMZ. If there is a Zone Map signed by a signature key, this descriptor MUST be present in the IEC62455 ECM [IEC62455] as described in Table 2. The field of this access criteria descriptor is defined in Table 3.

This descriptor signals the existence of a corresponding zone map, the zone map's identifier and the signature key. To comply with a MediaZones obligation, when this descriptor is included in IEC62455 ECM, a player application MUST locate the zone map which has the same zone map id as zone_map_id_ref in the descriptor. When this descriptor is not included in IEC62455 ECM, the MediaZones obligation is not applied to the content related to the IEC62455 ECM.

Table 2: DMZ access_criteria_descriptor

Access_criteria_descriptor	Tag	Comment
dmz	0x20	Reference to zoneMapId in zone map and encrypted signature key for zone map

Table 3: dmz access criteria descriptor

Field	No. of bits	Mnemonic
Zone_map_id_ref	16	uimsbf
encrypted_signature_key_material_length	8	uimsbf
for (i = 0; i < encrypted_signature_key_material_length; i++){		
encrypted_signature_key_material	8	bslbf
}		

zone_map_id_ref – ID of the zone map the DMZ Access Criteria Descriptor is referring to.

encrypted_signature_key_material_length – Length in byte of the encrypted signature key.

encrypted_signature_key_material – This filed contains the signature key that is used for signing the zone map referred to by zone_map_id_ref. The signature key is encrypted using AES-128-CBC, with fixed IV 0, and with 0 padding in the last block, if needed.

If the <programme_flag> == KSM_FLAG_TRUE, the signature key material is encrypted with the programme encryption key (PEK).

If <programme_flag> == KSM_FLAG_FALSE and <service_flag> == KSM_FLAG_TRUE, the signature key material is encrypted with the service encryption key (SEK).

7.2.3 Zone Map

When defining zone points for MPEG2-TS, the following mapping rules are applied:

- `accessUnitReference` field of the ZonePoint structure SHALL be a Presentation Time Stamp (PTS) in PES packet.
- `zone_map_id` SHALL be a unique identifier among zone maps signed by the same signature key to prevent a swapping of zone maps in the MPEG2-TS file.
- The zone map in the main media presentation SHALL be signed by the signature key conveyed in the ECM which includes the DMZ access criteria descriptor, Section 7.2.2, including `zone_map_id_ref` referring to the zone map.

The zone map for an audio and video elementary stream SHALL be included as private_section [13818-1] which SHALL have the PID value signaled as PID in the DMZ_Descriptor.
The following values SHALL be set in each field of the private_section for the zone map:

`table_id`: 0x98

`section_syntax_indicator`: 1

`private_indicator`: 0 (reserved)

`table_id_extension`: 0x00

`version_number`: 0

`private_data_byte`: binary encoding of the zone map (Appendix C). In case the binary encoding of the zone map exceeds the maximum size available for private data (4084 bytes), the zone map SHALL be split into blocks of 4084 or less byte that are embedded as private data with increasing `section_number` starting from 0.

In the zone map, following values SHALL be set in each field:

`mediaDigestAlgorithm`: 0 is set.

7.2.4 Zone Location

The zone location for an audio and video elementary stream SHALL be included as private_section [13818-1] which SHALL have the PID value signaled as PID in the DMZ_Descriptor.
The following values SHALL be set in fields the private_section for zone location:

`table_id`: 0x98

`section_syntax_indicator`: 1

`private_indicator`: 0 (reserved)

`table_id_extension`: 0x01

`version_number`: 0

`private_data_byte`: binary encoding of the zone location (Appendix D). In case the binary encoding of the zone location exceeds the maximum size available for private data (4084 bytes), the zone location SHALL be split into blocks of 4084 or less byte that are embedded as private data with increasing `section_number` starting from 0.

Appendix A Zone Map Binary Encoding for ISO Base Media File Format

For zone maps included as an ‘mZON’ box, the following syntax SHALL be used:

```
IsoMediaAccessUnit: {
    sample: unsigned int (32)
}
IsoMediaByteOffset: {
    sample: unsigned int (64)
}
InternalZoneInfo: {
  fromPoint: unsigned int (32)
  toPoint: unsigned int (32)
  id: unsigned int (32)
  attributes: unsigned int (8)
  mediaDigestAlgorithm: unsigned int (8)
  mediaDigestValue: {
    mediaDigestValueDataSize: unsigned int (8)
    mediaDigestValueData: bit (8) [mediaDigestValueDataSize]
  }
  meteringTag: {
    meteringTagDataSize: unsigned int (8)
    meteringTagData: utf-8-char (8) [meteringTagDataSize]
  }
}
ExternalZoneInfo: {
  splicePoint: unsigned int (32)
  id: unsigned int (32)
}
ZoneMap: {  points: {
    pointCount: unsigned int (32)
    pointElements: ZonePoint [pointsCount]
  }
  internalZones: {
    internalZoneCount: unsigned int (32)
    internalZoneElements: InternalZoneInfo [internalZoneCount]
  }
  externalZones: {
    externalZoneCount: unsigned int (32)
    externalZoneElements: ExternalZoneInfo [externalZoneCount]
  }
  signature: {
    signatureAlgorithm: unsigned int (8)
    signatureValue: {
        signatureValueDataSize: unsigned int (32)
        signatureValueData: bit (8) [signatureValueDataSize]
    }
  }
}
```

For zone maps included as an ‘mDMZ’ box, the following syntax SHALL be used:

```
IsoMediaAccessUnit: {
    sample: unsigned int (32)
}
IsoMediaByteOffset: {
    sample: unsigned int (64)
}
InternalZoneInfo: {
  fromPoint: unsigned int (32)
  toPoint: unsigned int (32)
  id: unsigned int (32)
  attributes: unsigned int (8)
  mediaDigestAlgorithm: unsigned int (8)
  mediaDigestValue: {
    mediaDigestValueDataSize: unsigned int (8)
    mediaDigestValueData: bit (8) [mediaDigestValueDataSize]
  }
  meteringTag: {
    meteringTagDataSize: unsigned int (8)
    meteringTagData: utf-8-char (8) [meteringTagDataSize]
  }
}
ExternalZoneInfo: {
  splicePoint: unsigned int (32)
  id: unsigned int (32)
}
Extension: {
   size:           unsigned int (32)
   type:           bit(32)
   criticalFlag: bit (8)
   payload:       bit (8) [size-9]
}
ZoneMap: {  points: {
    pointCount: unsigned int (32)
    pointElements: ZonePoint [pointsCount]
  }
  internalZones: {
    internalZoneCount: unsigned int (32)
    internalZoneElements: InternalZoneInfo [internalZoneCount]
  }
  externalZones: {
    externalZoneCount: unsigned int (32)
    externalZoneElements: ExternalZoneInfo [externalZoneCount]
  }
  extensions: {
    extensionCount: unsigned int (32)
    extensions: Extension [extensionCount]
  }
  signature: {
    signatureAlgorithm: unsigned int (8)
```

```
    signatureValue: {
        signatureValueDataSize: unsigned int (32)
        signatureValueData: bit (8) [signatureValueDataSize]
    }
  }
}
```

Appendix B Zone Location Binary Encoding for ISO Base Media File Format

```
SchemeLocationInfo: {
  scheme: unsigned int (32)
  locationValueSize: unsigned int (32)
  locationValue: bit (3) [locationValueSize]
}
ZoneLocationInfo: {
  id: unsigned int (32) // zone id
  schemeLocationCount: unsigned int (32)
  schemeLocationElements: SchemeLocationInfo[schemeLocationCount]
}
ZoneLocationMap: {
  zoneLocationCount: unsigned int (32)
  zoneLocationElements: ZoneLocationInfo [zoneLocationCount]
}
```

Appendix C Zone Map Binary Encoding for MPEG2-TS File Format

Syntax	No. of bits	Mnemonic
zone_map(){		
zone_map_id	16	uimsbf
zone_point_number	8	uimsbf
for(i=0;i<zone_point_number;i++){		
'0010'	4	bslbf
PTS[32..30]	3	bslbf
marker_bit	1	bslbf
PTS[29..15]	15	bslbf
marker_bit	1	bslbf
PTS[14..0]	15	bslbf
marker_bit	1	bslbf
}		
internal_zone_number	8	uimsbf
for(i=0;i<internal_zone_number;i++){		
from_point	8	uimsbf
to_point	8	uimsbf
id	8	uimsbf
attributes	8	uimsbf
media_digest_algorithm	8	uimsbf
media_digest_value_length	8	uimsbf
for(j=0;j<media_digest_value_length;j++){		
media_digest_value		
}	8	bslbf
metering_tag_length		
for(k=0;k<metering_tag_length;k++){	8	uimsbf
metering_tag		

}	8	bslbf
}		
external_zone_number		
for(i=0;i<external_zone_number;i++){	8	uimsbf
splice_point		
id	8	uimsbf
}	8	uimsbf
extension_count		
for(i=0;i<extension_count;i++){	8	uimsbf
type		
criticalFlag	8	uimsbf
size	32	uimsbf
for(j=0;j<size;j++){	16	uimsbf
payload		
}	8	bslbf
}		
signature_algorithm	8	uimsbf
signature_value_length	16	uimsbf
for(i=0;i<signature_value_length;i++){		
signature_value	8	bslbf
}		
}		

711

Appendix D Zone Location Binary Encoding for MPEG2-TS File Format

Syntax	No. of bits	Mnemonic
zone_location_map(){		
zone_location_number	8	uimsbf
for(i=0;i<zone_location_number;i++){		
id	8	uimsbf
scheme_location_number	8	uimsbf
for(j=0;j<scheme_location_number;j++){		
scheme	32	uimsbf
location_value_length	8	uimsbf
for(k=0;k<location_value_length;k++){		
location_value	8	bslbf
}		
}		
}		
}		

Appendix E Examples (Informative)

E.1 FBI Warning

In this example, the media presentation starts with an FBI Warning screen that must be viewed before the rest of the presentation, unless that screen has already been viewed in the past 30 days.

```
ZoneMap: {
  points: [AU-1, AU-189]
  internalZones: [
    {
      fromPoint=0,
      toPoint=1,
      mediaDigestAlgorithm=1,
      mediaDigest=[…],
      id=100,
      attributes=1,
      meteringTag=""
    }
  ]
  externalZones: []
  signature: {
      signatureAlgorithm=0,
      signatureValue=[…]
  }
```

This zone map contains 2 points and one zone. The Warning zone spans access unit 1 to access unit 189 and is marked as INSERTED (attributes=1), which indicates to a player that the zone has been inserted into the main presentation and may be skipped (starting play directly at access unit 189), unless the zone has a NOSKIP obligation returned by the Perform method of the Play action.

The pseudo-code for the Control would be:

```
ESB-1 = {
  ACTION_GRANTED
  Obligations:
    MediaZones: [[100, STICKY, 0], [100, NOSKIP, 0]]
  Callbacks:
    OnZoneCompleted: [100, (RESET, ZoneCallback, 0)]
}

ESB-2 = {
  ACTION_GRANTED
}
Control.Actions.Play.Perform:
```

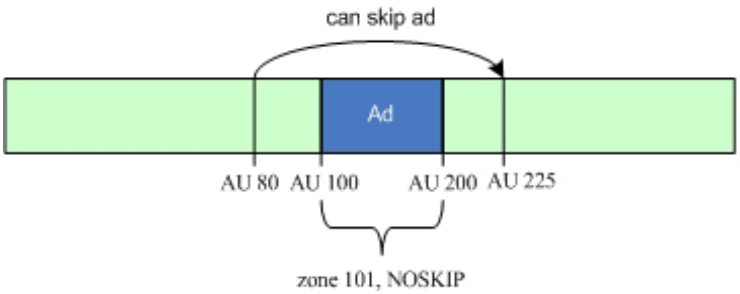
```
lastViewed =
GetHostObject(/Octopus/SeaShell/Databases/Marlin/ACME/Zones/movie
-0007-warning/lastViewed)
if now-lastViewed > 30 days then:
    return ESB-1 // the zone 100 is NOSKIP and STICKY
else:
    return ESB-2 // you can skip the warning

ZoneCallback:
now = GetTrustedTime()
SetHostObject(/Octopus/SeaShell/Databases/Marlin/ACME/Zones/movie
-0007-warning/lastViewed, now)
return ESB-2 // you can now skip the warning
```

As you can see from the pseudocode above, the Control for the Perform method of the Play action gets the value of the Seashell database object /Octopus/SeaShell/Databases/Marlin/ACME/Zones/movie-0007-warning/lastViewed and compares it with the current time. If the difference is greater than 30 days, the Control returns ESB-1, which specifies that zone 100 has a STICKY obligation and a NOSKIP obligation. ESB-1 also specifies that a callback, ZoneCallback, must be called after play of zone 100 is complete. If the time difference is less than 30 days, the Control returns ESB-2, which does not specify any obligations or callbacks.

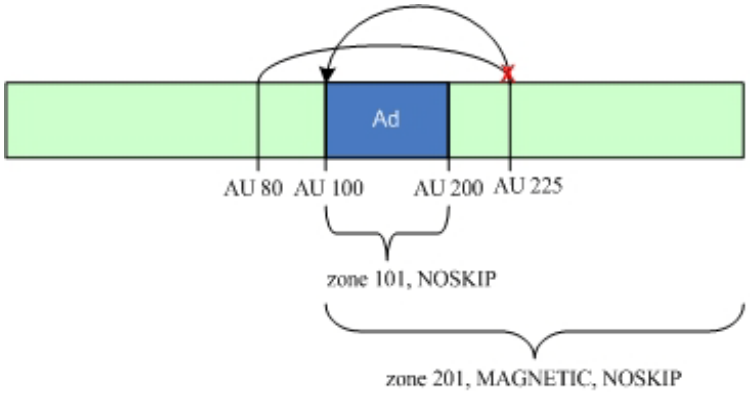
If ESB-1 is returned, then the FBI warning (zone 100) is STICKY and NOSKIP and, after playback is complete, ZoneCallback is invoked. It gets the value of the current time, and stores that value in the /Octopus/SeaShell/Databases/Marlin/ACME/Zones/movie-0007-warning/lastViewed object. It then returns ESB-2. Since ZoneCallback is of type RESET, any obligations and callbacks it returns (including zero) must replace any previous obligations and callbacks. In this example, what is specified in ESB -2 (no obligations or callbacks) replaces what was specified in ESB-1, so the FBI warning does not have to be viewed again until a subsequent Play action Control time comparison shows a difference of greater than 30 days and thus once again returns ESB-1.

E.2 Preventing Skipping of a Zone

Consider the situation illustrated in the following diagram, showing a NOSKIP zone containing an advertisement. The intent is for the advertisement to have to be played. However, specifying a NOSKIP obligation for the zone is not sufficient. The NOSKIP obligation ensures that the player itself will not skip the zone, once it reaches the zone, but that obligation does not prevent skipping over the zone starting from a point before the zone.



The way to ensure that the ad is shown is to also have a MAGNETIC obligation (and a NOSKIP obligation) for a zone starting at the beginning of the advertisement and ending at the end of the media presentation. When a seek is done to position inside a MAGNETIC zone, subsequent zone playback does not start at the position seeked to, but rather starts at the very beginning of the MAGNETIC zone. See the following diagram:



Now an attempt to seek from any position before the ad (such as access unit 80) to any position beyond the ad (such as access unit 225) will not be successful. The MAGNETIC obligation causes the seek attempt to result in subsequent playback starting at the beginning of the MAGNETIC zone, which in this case is also the beginning of the ad.

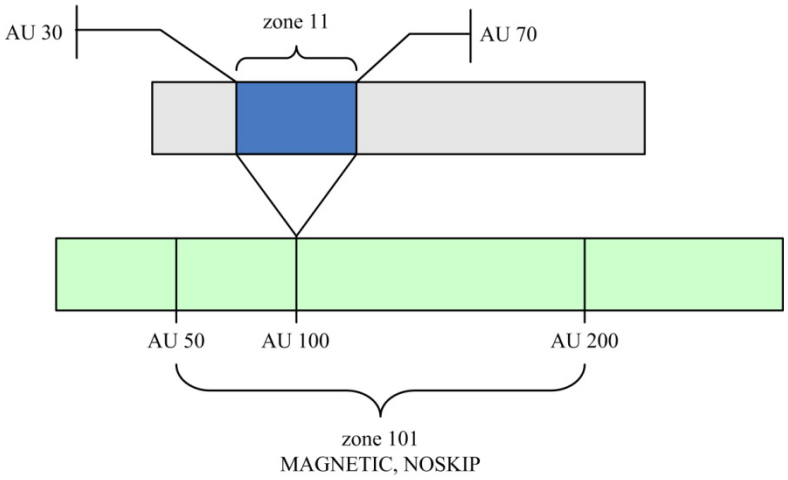
As with the example in §E.1, the Control program for this example would be likely to specify a callback method to be called after play of zone 101 is complete. That callback method could, for example, eliminate the obligations for zones 101 and 201, by being a RESET callback that does not return any obligations. Once that is done, the ad does not have to be viewed again, and skipping from any position within the main media presentation to any other position is possible.

E.3 External Zones and Inherited Obligations

There are various reasons it might be desirable to have some of the media for a presentation in one file and other portions of the media in a different file. For example, you may want to have the main presentation in one file, and advertisements in another file, so you do not have to repackage the entire content each time you change the ads. In this case, you would typically splice the advertisements at various points in the main presentation, using external zones.

An external zone spliced into a media presentation is considered to be included in the span of media of the zone into which it is spliced, so it inherits the obligations of the zone into which it is spliced. The diagram below shows a zone (zone 101) that spans from access unit 50 to access unit 200. A zone (zone 11) from a different file is specified as an external zone to be spliced at access unit 100.

Any external zone spliced into a point within zone 101 is treated as a part of zone 101, just as though the media within the spliced-in zone actually appeared directly within zone 101. In this case, zone 101 has a MAGNETIC obligation, so any attempt to seek into zone 11 will not be successful; playback will resume at access unit 50 (the start of zone 101), the same as would occur if any other attempt to seek into zone 101 was made.



Here is the zone map for the file containing zone 101:

```
ZoneMap: {
  points: [AU-50, AU-100, AU-200]
  internalZones: [
    {
      fromPoint=0,
      toPoint=2,
      mediaDigestAlgorithm=1,
      mediaDigest=[…],
      id=101,
      attributes=0,
      meteringTag=""
    }
  ]
  externalZones: [
      splicePoint=1,
      id=11
  ]
  signature: {
      signatureAlgorithm=0,
      signatureValue=[…]
  }
```

Here is the zone map for the file containing zone 11:

```
ZoneMap: {
```

```
  points: [AU-30, AU-70]
  internalZones: [
    {
      fromPoint=0,
      toPoint=1,
      mediaDigestAlgorithm=1,
      mediaDigest=[…],
      id=11,
      attributes=1,
      meteringTag=""
    }
  ]
  externalZones: []
  signature: {
      signatureAlgorithm=0,
      signatureValue=[…]
  }
```

E.4 INCLUDE_SPLICE Usage

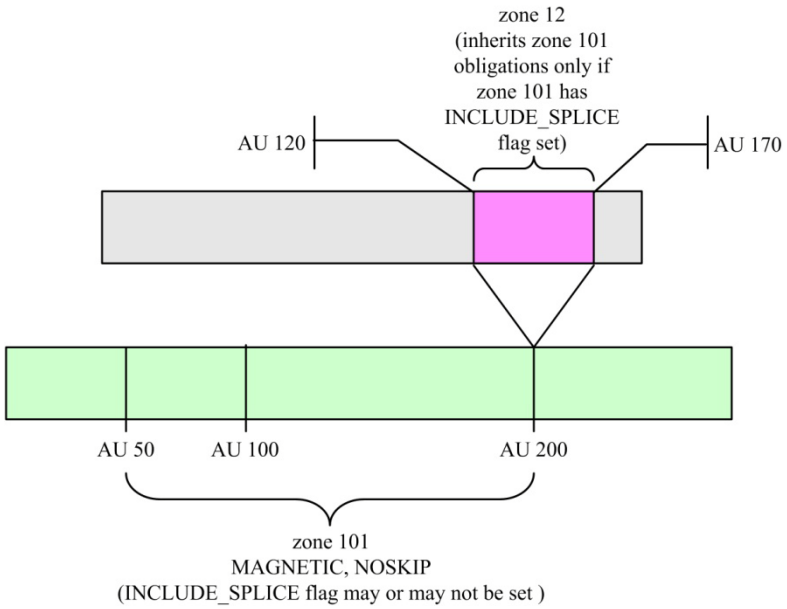
As previously stated, and illustrated in §E.3, any external zone spliced into a point within another zone inherits the obligations of the zone into which it is spliced. The media considered to be in a zone is the media beginning at the fromPoint on up to, but not including, the media specified at the toPoint. An external zone spliced in at the beginning of a zone (the fromPoint) or at any point between there and the end of the zone is considered a part of the zone into which it is spliced, so it inherits the obligations of the zone into which it is spliced.

If the splicePoint is the same as the toPoint of a zone, the spliced zone may or may not be considered a part of the zone preceding it, that is, the zone whose toPoint is the same as the splicePoint.

In this situation, if the zone preceding the spliced-in zone has an obligation with the INCLUDE_SPLICE flag set, the zone spliced in is considered part of the zone into which it is spliced, and it inherits any obligations (MAGNETIC and NOSKIP in the diagram below) of that zone.

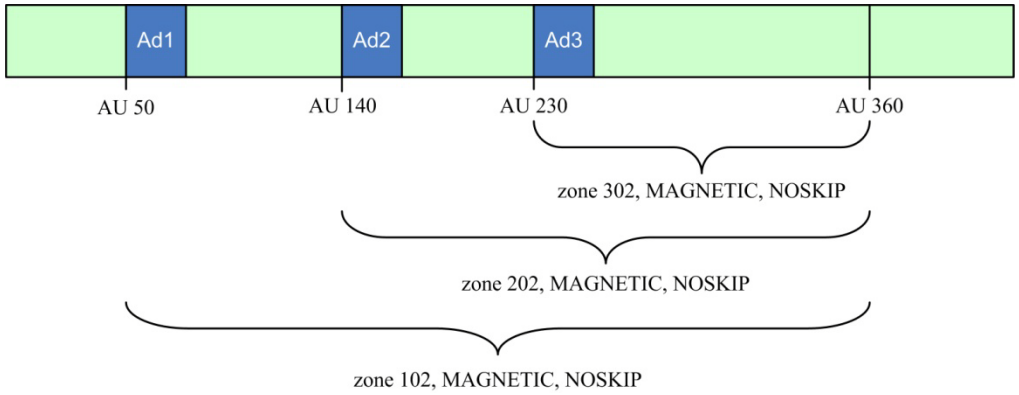
If the zone preceding the spliced-in zone *does not* have an obligation with the INCLUDE_SPLICE flag set, the zone spliced in is *not* considered part of the zone into which it is spliced, so it does not inherit any obligations.

The following diagram illustrates the situation, with external zone 12 being spliced at access unit 200, which is the toPoint of zone 101:



E.5 Overlapping Zones

This example illustrates how, when the media player application attempts to seek to a position that is inside more than one MAGNETIC zone, the playback must begin at the earliest fromPoint of all those zones. The following diagram illustrates three overlapping zones:



Zone 102 spans access unit 50 to access unit 360, zone 202 spans access unit 140 to access unit 360, and zone 302 spans access unit 230 to access unit 360. Each zone begins with an advertisement, and the content provider would like to ensure that playback of each advertisement is at least started. (For simplification purposes, the fact that each ad is in a NOSKIP zone is not illustrated.)

If a seek is done to any position between access unit 50 and access unit 359 (inclusive), playback will begin at access unit 50, since that is the earliest fromPoint of the overlapping MAGNETIC zones.

If, for example, a seek is done from a position prior to access unit 50 to a position between access units 50 and 139 (inclusive), then the destination point is within zone 102 and the MAGNETIC obligation of that zone ensures that playback begins at the beginning of that zone, access unit 50. If instead a seek is done from a position prior to access unit 50 to a position between access units 140 and 229 (inclusive), then the destination point is within both zone 102 and zone 202. Just as in the previous case, playback begins at access unit 50, because a seek to a position inside more than one MAGNETIC zone requires playback to begin at the earliest fromPoint of those zones, which in this case is access unit 50.

As with other examples, the Control program for this example would be likely to specify a callback method to be called after play of each ad is complete. The callback method executed after Ad1 has been played could, for example, eliminate the obligations for that ad's zone and for zone 102. Once that is done, the ad does not have to be viewed again, and zone 102 would no longer be MAGNETIC, so a seek could be done to any position prior to access unit 140 and playback would resume at the position seeked to. At this point, a seek to a position anywhere between access units 140 and 359 (inclusive) would result in playback starting at access unit 140, the earliest fromPoint of the MAGNETIC zones within that range.

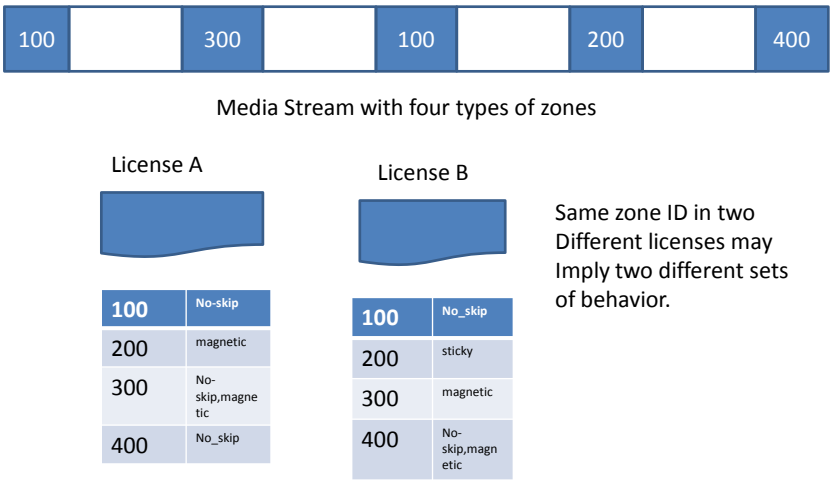
E.6 Zone Id as a Behavioral Id

This example shows how zone identifiers can be used as behavior identifiers. The usage of zone identifiers as behavioral identifiers facilitates the definition of zone behaviors in advance in a Control of a license. The service providers can then subsequently map ids to a certain behavior in a Marlin license. The service provider can issue a new license only if he wants to introduce a new obligation zone.

ID1	100
ID2	200
ID3	300
ID4	400

The above diagram illustrates four pre-defined media zone ids. The service provider can then define licenses to map these ids to a certain zone behavior in a license. The

diagram below illustrates two types of media zone behaviors accomplished through two different types of licenses:



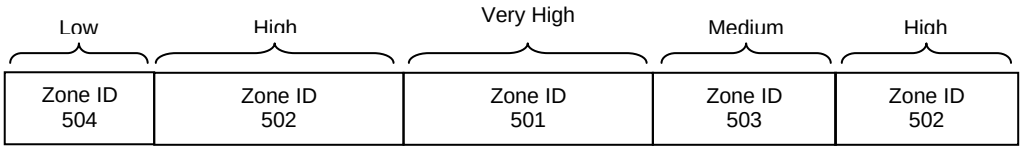
100	No-skip
200	magnetic
300	No-skip,magnetic
400	No_skip

100	No_skip
200	sticky
300	magnetic
400	No-skip,magnetic

The subscribers can be issued licenses in advance before the streaming of the media begins. For instance, a media player of a subscriber who has been issued License A will enforce obligations as per mapping in the License A. Similarly another subscriber who has been issued the License B will enforce the obligations as per mapping in the License B. Note that the media stream carries four types of zones. A service provider can enforce a different set of behavior by issuing a new license. Also, a service provider can define additional zone ids and issue new licenses.

E.7 Differentiated Services using Subscription Level and Commercial value (priority) of an inserted media content (Ad)

This example illustrates application of DMZ specifications to a use case where a service provider dynamically assigns priorities to inserted media content (or ad) based on its commercial value—Very High, High, Medium and Low. Similarly, we can have subscribers with different subscription levels—Platinum, Gold, Silver and Bronze. The subscription level is set in the Subscription Node Object when a subscriber requests a service of a particular subscription level. A media player may then enforce decision regarding skipping an ad based on the subscription level of the user and commercial value (priority) of an ad. For instance, a subscriber with the Gold level subscription will be allowed to skip Medium and Low priority ads but he would not be able to skip ads marked High & Very High Priority in the media stream. Similarly a subscriber with priority level corresponding to Low (Bronze Subscribers) will not be able to skip Ads of any priority.



The pseudo-code for the Control would be:

```
//Platinum subscriber can skip all zones except the zone marked
//Very High priority (behavioral id=501)

ESB-Platinum = {
  ACTION_GRANTED
  Obligations:
    MediaZones: [[501, NoSkip, 0]]
}

//Gold subscriber can skip all zones except the zones marked
//Very High priority (behavioral id=501 and High Priority
//(behavioral id=502)

ESB-Gold = {
  ACTION_GRANTED
  Obligations:
    MediaZones: [[501, NoSkip, 0], [502, NoSkip, 0]]
}


//Silver subscriber can't skip zones marked Very High
//priority (behavioral id=501, High Priority (behavioral
//id=502)and Medium priority (behavioral id=503)

ESB-Silver = {
  ACTION_GRANTED
  Obligations:
    MediaZones: [[501, NoSkip, 0], [502, NoSkip, 0], [503,
NoSkip, 0]
}

// Bronze subscriber can't kip zones marked Very High priority,
//High priority, Medium priority and Low priority (behavioral
//id=504)

ESB-Bronze = {
  ACTION_GRANTED
  Obligations:
    MediaZones: [[501, NoSkip, 0], [502, NoSkip, 0], [503,
NoSkip, 0], [504, NoSkip, 0]]
}
```

Control.Actions.Play.Perform:

```
If (Node-reachable==Platinum)
      Return ESB-Platinum
Else If (Node-reachable==Gold)
      Return ESB-Gold
Else If (Node-reachable==Silver)
      Return ESB-Silver
Else if (Node-reachable==Bronze)
      Return ESB-Bronze
```