MICROSOFT CORPORATION

Content Decryption Module Interface Specification

An open interface for enabling HTML5 Encrypted Media Extensions in open source browsers

John C. Simmons
January 2, 2014

Version 1.0

Abstract:
The W3C HTML working group is developing media extension specifications for HTML5 to enable the delivery of commercial video to consumers over the Web. One of these is the Encrypted Media Extensions (EME) specification. The current specification describes an open interface which may be used to implement an EME-compliant Content Decryption Module (CDM) within a User-agent, providing access to a platform DRM component which supports the defined Content Decryption Module interface (CDMi).

Legal Notice

Contents

1 Introduction 1

1.1 Scope 1
1.2 Conventions 1
1.3 Terminology, Abbreviations and Acronyms 2
1.4 References 4
1.5 Revision History 5

2 Content Decryption Module Interface 6

2.1 Architecture 6
2.2 CDM Interface Object Model 7
2.3 Cdm_MediaKeys Object 8
2.4 Cdm_MediaKeySession Object 9

Figures

Figure 1 Content Decryption Module Interface Entity Relationship Diagram 6
Figure 2 CDM Interface Object Model 7

Tables

Table 1 CDMi Implementation ReadyState 9

CONTENT DECRYPTION MODULE INTERFACE SPECIFICATION

VERSION 1.0

JANUARY 2, 2014

1 INTRODUCTION

Recent standardization advances have significantly enhanced the interoperability of commercial Web media services [DRM]. The W3C HTML Working Group is developing HTML Media Extensions for the support of these services. As of the time of this writing, there are two principle HTML Media Extensions under development – the Media Source Extensions [MSE] and the Encrypted Media Extensions [EME][1].

Microsoft has developed a generalized interface enabling open source browsers to support Encrypted Media Extensions – the Content Decryption Module interface (CDMi).[2]

1.1 SCOPE

An open interface for accessing a platform DRM Content Decryption Module (CDM) which may be used by a User-agent to expose CDM functionality as specified by the W3C HTML Encrypted Media Extensions.

1.2 CONVENTIONS

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

- “MUST”, “REQUIRED” and “SHALL” mean that the definition is an absolute requirement of the specification.
- “MUST NOT” and “SHALL NOT” mean that the definition is an absolute prohibition of the specification.

[1] There are two additional W3C specifications outside the media extensions activity which will become important for consumer web distribution -“Web Cryptography API” and “Web Crypto Key Discovery”.
[2] Microsoft will make available to PlayReady licensees a CDMi Implementation designed to work with the PlayReady Device Porting Kit (Device PK).

- "SHOULD" and "RECOMMENDED" mean that there may be valid reasons to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- "SHOULD NOT" and "NOT RECOMMENDED" mean that there may be valid reasons when the particular behavior is acceptable, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- "MAY" and "OPTIONAL" means the item is truly optional.

1.3 TERMINOLOGY, ABBREVIATIONS AND ACRONYMS

1.3.1 TERMINOLOGY

AppData	Application specific data which is passed in some applications with the InitData when creating a Key Session.[3]
Content Decryption Module (CDM)	Component of the User-agent which provides support for one or more Key Systems [EME]. It is transparent to the Web application whether a CDM is part of or separate from the User-agent. A User-agent can support multiple CDMs.
Content Decryption Module interface (CDMi)	An open, interoperable interface enabling a User-agent to implement CDM functionality as provided by a platform DRM.
CDMi Implementation	DRM-specific software which exposes the CDM interface, providing a translation between EME methods and events to the equivalent functions of the underlying platform DRM.
CDMi Implementation License Store	Optional local License store maintained by the CDMi Implementation.
Encrypted Media Extensions (EME)	An HTML Media Extensions specification which extends the HTMLMediaElement to enable playback of Protected Resources.
Globally Unique Identifier (GUID)	A unique reference number, represented as a 32-character hexadecimal string, and usually stored as a 128-bit value.

[3] The AppData field is not yet in the official EME specification but is implemented in IE11 and will be proposed by Microsoft to the EME specification.

HTML Media Extensions	A suite of specifications either proposed, under development or completed in the W3C HTML working group providing added browser functionality for commercial web video services; e.g. Media Source Extensions [MSE] and Encrypted Media Extensions [EME].
Initialization Data	In HTML EME, a generic term for container-specific data that is used by CDMs to generate a Key Request [EME] message.
Key	Decryption key provided in a DRM License.
Key Request message	In HTML EME, a generic term for the Key or License acquisition message sent to the License Server on behalf of the CDM [EME].
Key Session	Interchange between the JavaScript application and the CDMi Implementation for acquisition of the Key or Keys needed to decrypt media. See Media Session.
Key System	In EME, a generic term for a decryption mechanism and/or content protection provider [EME].
Key System String	A reverse domain name identifying the Key System [EME]. For example, the Microsoft PlayReady Key System String is "`com.microsoft.playready`".
keyadded event	An HTML EME event indicating a Key has been added as the result of an `update` call [EME].
keyerror event	An HTML EME event indicating an error has occurred in one of the HTML EME methods or in the CDMi Implementation [EME].
keymessage event	An HTML EME event indicating a message has been generated, most likely one which must be sent to a License Server [EME].
License	A DRM data structure that includes policies and an encrypted content Key.
License Server	A server which provides DRM Licenses to clients.
Licensed Product	The components of an implementation which are subject to the DRM Provider Compliance and Robustness Rules.
Media Session	The interchange between the media engine and the CDMi Implementation for the decryption of media samples. This interchange is DRM-specific, is subject to the DRM provider compliance and robustness rules, and is not documented in this specification (see section 2.1, below).

Media Source Extensions (MSE)	An HTML Media Extensions specification which extends the HTMLMediaElement to facilitate use cases like adaptive streaming and time shifted live streams.
Protection System Specific Header box ('pssh')	In the ISO Base Media File Format, the Protection System Specific Header box contains Initialization Data needed by a specific content protection system to decrypt the media content [ISOBFF].
Session ID	In HTML EME, a session ID is an optional string ID used to associate calls related to a Key/License lifetime, starting with the Key Request message [EME]. See Key Session.
Update	(EME session method) provides the License Server response of a keymessage to the CDM Key System [EME].
User-agent (UA)	The client software used by an End-user to access HTTP servers.

1.3.2 ABBREVIATIONS AND ACRONYMS

CDM	Content Decryption Module
CDMi	Content Decryption Module interface
EME	Encrypted Media Extensions
GUID	Globally Unique Identifier
JS	JavaScript
KID	Key Identifier
MSE	Media Source Extensions
UA	User-agent

1.4 REFERENCES

1.4.1 NORMATIVE REFERENCES

[EME] *"Encrypted Media Extensions",* http://www.w3.org/TR/encrypted-media/ (*latest editors draft:* https://dvcs.w3.org/hg/html-media/raw-file/default/encrypted-media/encrypted-media.html)

[HTML5] "*HTML5 A vocabulary and associated APIs for HTML and XHTML*", *editors draft*, http://dev.w3.org/html5/spec

[RFC2119] "*Key words for use in RFCs to Indicate Requirement Levels*", S. Bradner, March 1997, http://www.ietf.org/rfc/rfc2119

1.4.2 INFORMATIONAL REFERENCES

[CENC] ISO/IEC 23001-7: 2011, "*Information technology – MPEG systems technologies – Part 7: Common encryption in ISO base media file format files*".

[CRYPTO] "*Web Cryptography API*", http://www.w3.org/TR/WebCryptoAPI

[DASH] ISO/IEC 23009-1:2012, "*Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*", http://standards.iso.org/ittf/PubliclyAvailableStandards/c057623_ISO_IEC_23009-1_2012.zip

[DRM] "*Interoperability, Digital Rights Management and the Web*", John C. Simmons, Microsoft Corp., Dr. Stefan Arbanowski, Fraunhofer FOKUS Research Institute, http://www.microsoft.com/playready/documents/

[ISOBFF] ISO/IEC 14496-12, Third Edition, "*Information technology – Coding of audio-visual objects – Part 12: ISO Base Media File Format*", with Corrigendum 1:2008-12-01, Corrigendum 2:2009-05-01, Amendment 1:2009-11-15 and Amendment 3:2011-08-17.

[KEYDSC] "*WebCrypto Key Discovery*", http://www.w3.org/TR/webcrypto-key-discovery

[MSE] *"Media Source Extensions",* http://www.w3.org/TR/media-source/

1.5 REVISION HISTORY

Version 1.0 Initial version
2-Jan-2014

2 CONTENT DECRYPTION MODULE INTERFACE

The Content Decryption Module interface (CDMi) is an open, interoperable interface enabling a User-agent to implement CDM functionality as provided by a platform DRM. This interface is exposed by a CDMi Implementation, providing a translation between HTML Encrypted Media Extension (EME) methods and events to the equivalent functions of the underlying platform DRM.

The CDMi object model parallels that of the MediaKeys and MediaKeySession objects in the W3C Encrypted Media Extensions specification [EME] and is intended to have no DRM-provider specific elements.

2.1 ARCHITECTURE

The relationship of the Content Decryption Module interface Implementation to the browser, the media engine and the underlying Platform DRM are shown in the Figure 1.

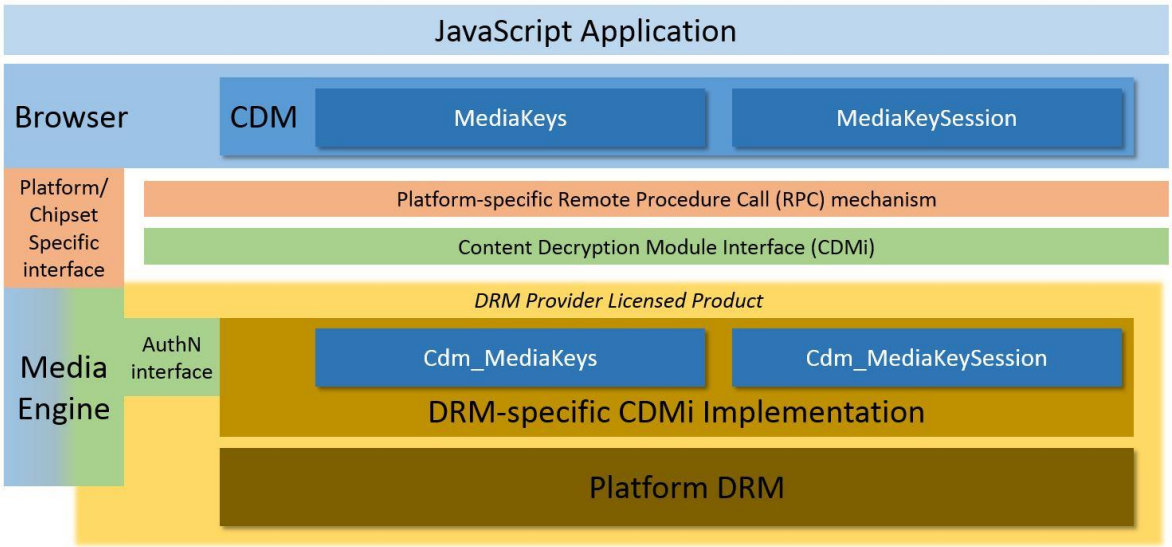

Figure 1 Content Decryption Module Interface Entity Relationship Diagram

The **CDMi Implementation** is part of the DRM provider's **Licensed Product** – i.e. subject to the compliance and robustness rules of the DRM provider's license agreement. It is this software which contains the actual CDM objects used for Key acquisition.

The **CDM** MediaKeys and MediaKeySession objects in the browser are a remote procedure call 'projection' from the CDMi Implementation, using **a platform-specific RPC mechanism**.

The **Content Decryption Module interface (CDMi)** is a Microsoft published open interface specification; browsers can utilize this interface without being a Licensed Product.[4]

A **media engine** performs the decoding of the protected audio or video elementary streams outside the browser. In practice, this can be secure, hardware assisted decoding. The interface between the browser and the media engine is platform and chipset specific and outside the scope of this specification.

The media engine uses an **authenticated interface** to establish a Media Session for communicate to the CDMI Implementation. This ensures that only a DRM Provider-trusted media engine MAY pass media samples to the CDMi Implementation for decryption. This Media Session is DRM-specific, is not implemented by the User-agent, and is therefore not documented in this open interface specification.[5]

The CDMi Implementation contains the actual CDM functionality, enabling the browser CDM to expose the methods and events defined by the W3C Encrypted Media Extensions without compromising content protection or requiring DRM-specific functionality to be implemented within the browser.

2.2 CDM INTERFACE OBJECT MODEL

The CDM Interface is an object modeled after the EME `MediaKeys` object. Figure 2 summarizes the CDMi object model.

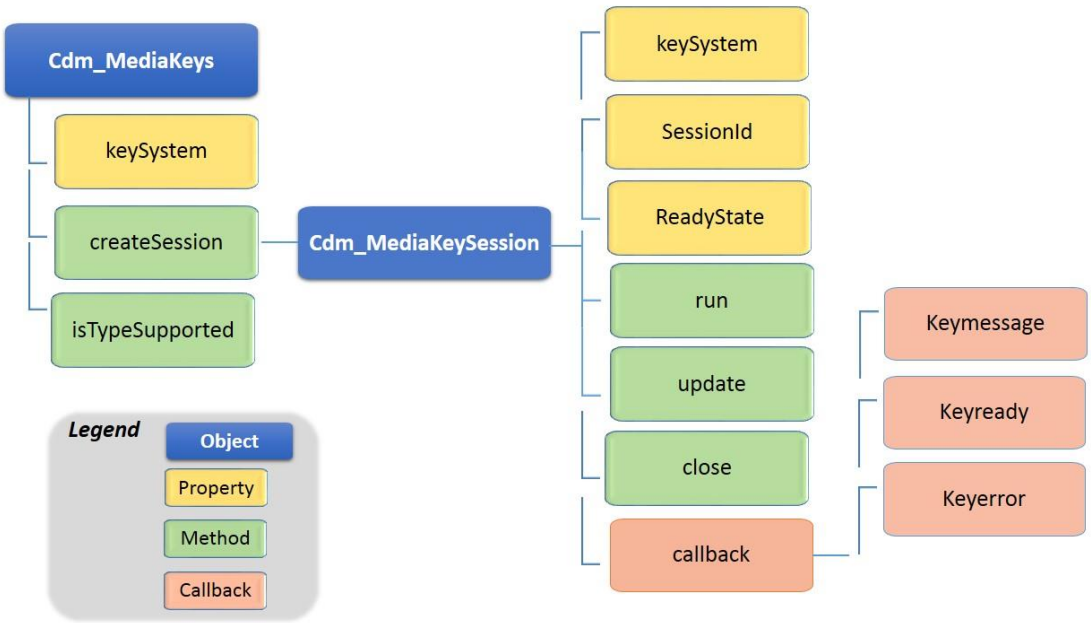

Figure 2 CDM Interface Object Model

[4] Microsoft will provide to PlayReady licensees a CDMi Implementation on top of the PlayReady Device Porting Kit.
[5] For example, the PlayReady CDMi Implementation incorporates a Media Session interface, to be used by the PlayReady licensee in the platform to secure content decryption.

2.3 CDM_MEDIAKEYS OBJECT

The `Cdm_MediaKeys(wchar_t *keySystem)` constructor MUST run the following steps:

1. If the `keySystem` is null, an empty string, or not the keySystem identifier for this specific CDMi Implementation - fail.[6]

2.3.1 ATTRIBUTE KEYSYSTEM

The `Keysystem` attribute is a `wchar_t *` pointer to an identifier of the Key System being used [EME].[7]

2.3.2 CREATESESSION METHOD

Creates a CDM Key management session.

Use: `createSession( wchar_t * type, const unsigned char *initData, const unsigned char *AppData)`

`type`	MIMETYPE of the media. This MAY be ignored.
`initData`	As defined in [EME].
`AppData`	CDM specific data which is passed in some applications with the InitData.[8]

Algorithm

1. The CDMi Implementation creates a new `Cdm_MediaKeySession` object with a ready state of `START`
2. The CDMi Implementation sets the `Cdm_MediaKeySession SessionId` to a random value – a base 64 encoded GUID.
3. The `InitData` and `AppData` are parsed and the CDMi Implementation specific data is extracted from the InitData[9].
4. Initialize the DRM Key Session state. This action is CDMi Implementation specific.

[6] Even though Cdm_Init is a CDMi Implementation specific call, it takes the keySystem parameter because the reverse domain name format is intended to handle versioning, as well.
[7] For the Microsoft PlayReady CDMi Implementation, this is "com.microsoft.playready".
[8] The AppData field is not yet in the official EME specification but is implemented in IE11 and has been proposed by Microsoft to the EME specification. It enables Application specific information to be carried securely with the license request.
[9] For the ISO Base Media File Format, the CDMi Implementation specific information extracted from initData will be a Protection System Specific Header box [CENC].

2.3.3 ISTYPESUPPORTED METHOD

Determines whether a media type is supported for decryption by the CDMi Implementation.

```
BOOL IsTypeSupported(wchar_t *Keysys, wchar_t *type )
```

Keysys	Identifier of the Key System being used [EME].
type	Pointer to a string characterizing the MIMETYPE of the media.

Algorithm

1. Return TRUE if `Keysys` string and `type` are both supported by the CDMi Implementation.

2.4 CDM_MEDIAKEYSESSION OBJECT

Key management session object created by `Cdm_MediaKeys createSession` method.

2.4.1 ATTRIBUTE KEYSYSTEM

The `Keysystem` attribute is a `wchar_t *` pointer to an identifier of the Key System being used [EME].

2.4.2 ATTRIBUTE SESSIONID

A `wchar_t *` pointer to a unique Key Session identifier.

This CDMi Implementation creates a randomly generated unique session ID (based 64 encoded GUID). It is fixed for the session after it is generated.

2.4.3 ATTRIBUTE READYSTATE

An enum indicating the readyState of the CDMi Implementation.

```
enum ReadyState {
   "start",
   "pending",
   "ready",
   "closed",
   "error"
};
```

Table 1 CDMi Implementation ReadyState

Enumeration	Description
start	Indicates the MediaKeySession has been created but has not yet generated a keymessage.
pending	The MediaKeySession has generated a keymessage to acquire a Key, but has not successfully received the Key.
ready	The MediaKeySession has received a Key for decrypting the content.

Enumeration	Description
closed	A closed event has been fired at the MediaKeySession
error	An error has occurred and the Key Session has failed.

2.4.4 MEDIAKEYSESSION::RUN METHOD

Once the media Key Session has been established, a Keymessage can be generated, if needed.

Use: `long MediaKeySession::Run(callback);`

`callback` `Session Callback` object (see 2.4.7, below).

Algorithm

1. If the CDMi Implementation supports locally cached Licenses, and a License exists in the CDMi Implementation License Store which can be used with this Key Session:
    a. The CDMi Implementation will use the `Session Callback` object `Keyready` method to notify the UA CDM that the Key is ready for us.
2. Else
    a. The CDMi Implementation will generate a License challenge to be sent to its License Server
    b. If successful:
        - The CDMi Implementation will use the `Keymessage` method to prompt the UA CDM to notify the JS application to issue a License Request message to the specified License Server
        - The CDMi Key Session `readyState` is changed to `PENDING`.
    c. Else
        - The CDMi Implementation will use the `Session Callback` object `keyerror` method to notify the UA CDM of the error.

2.4.5 UPDATE METHOD

This method is called by the UA CDM to process a License response.

Use: `long Update(const unsigned char *key, unsigned long cb);`

`key` ..... Pointer to an array of bytes containing the response from the License Server.
`cb` ..... Count byte

Algorithm

1. The CDMi Implementation process the response from the License Server.

2. If successful:
    a. The CDMi Implementation will use the `Session Callback` object `keyready` method to inform the UA CDM that the Key has been added.
3. Else
    a. The CDMi Implementation will use the `Session Callback` object `keyerror` method to inform the UA CDM of the error.

2.4.6 CLOSE METHOD

Used by the UA CDM to close the CDMi Key Session.

Use: `void Close();`

Algorithm

1. The CDMi Implementation closes the CDMi Key Session.

2.4.7 SESSION CALLBACK OBJECT

The methods in the Callback object are provided by the UA CDM so that the CDMi Implementation can provide information needed for the [EME] `keyready`, `keyerror` and `keymessage` events.

2.4.7.1 KEYMESSAGE METHOD

This is a message generated by the CDMi Implementation, passed to the UA CDM and exposed to the JS Application as a `keymessage` event. It is likely a message to be sent to the License Server.

Use: `void Keymessage (unsigned int *mesg, unsigned long cbmesg, char *dest, unsigned long cbdest);`

`mesg`	Pointer to an array of bytes containing the message
`cbmesg`	Count byte of message
`dest`	URL to send the message.
`cbdest`	Count byte of destination.

Algorithm

1. The CDMi Implementation will use a DRM specific mechanism to construct the Key Message.

2.4.7.2 KEYREADY METHOD

This method informs the UA CDM that the Key is ready for use, either because the CDMi Implementation found an appropriate License in the CDMi Implementation License Store (see `CreateSession` method, above) or because the License was successfully received from the License Server (see `Update` method, above).

Use: `void Keyready();`

2.4.7.3 KEYERROR METHOD

This method informs the UA CDM that an error has occurred in the CDMi Implementation with regards to the present Key Session.

Use: void Keyerror (unsigned short error, unsigned long syserror)

`error`	The error value
`syserror`	The system error value

