Marlin Broadband Delivery System Specification

Version 1.2.3
Final

Source Marlin Developer Community
Date April 27, 2010

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

1.1 Document Organization

This specification defines the Marlin Broadband Delivery System. It contains:

- (This) introduction, including abbreviations, definitions, and references.
- An overview of the Marlin Broadband Delivery System and its relationship to the Marlin Core System Specification.
- Normative elements this specification introduces over and above those of the Marlin Core System Specification. These elements include:
    - Roles
    - Domain Policies
    - Protocols
    - Usage rules
- A set of appendices containing the XML schemas and WSDLs for Marlin Broadband Services.

1.2 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 RFC 2119 [RFC2119].

These capitalized key words are used to unambiguously specify requirements and behavior that affect the interoperability and security of implementations. When these key words are not capitalized they are meant in their natural-language sense.

All elements of this specification are considered Normative unless specifically marked Informative. All Normative Elements are Mandatory to implement, except where such an element is specifically marked OPTIONAL. Finally, where Normative elements are described as OPTIONAL, they MAY be omitted from an implementation, but when implemented, they MUST be implemented as described.

1.3 Namespaces and Identifiers

This specification defines schemas conforming to XML Schemas [Schema] and normative text to describe the syntax and semantics of XML-encoded objects and protocol messages. In cases of disagreement between the schema documents and the schema listings in this specification, the schema documents take precedence. Note that in some cases the normative text of this specification imposes constraints beyond those indicated by the schema documents.

1.3.1 Namespaces and Notation

The following table summarizes the normative schemas defined by this specification, and their XML namespace [XMLns] URIs. These URIs MUST be used by implementations of this specification:

Prefix	XML Namespace	Schema File Name	Description

bbexc:	urn:marlin:broadband:1-1:nemo:services:schemas:exceptions		Broadband exception values
bsa:	urn:marlin:broadband:1-2:nemo:services:action-token	Broadband-services-action.xsd	Action Token schema
bsc:	urn:marlin:broadband:1-2:nemo:services:configuration	Broadband-services-config.xsd	Configuration Token schema
dcs:	urn:marlin:broadband:1-1:nemo:services:schemas:data-certification-service	DataCertification.xsd	Data Certification Service schema
dcsi:	urn:marlin:broadband:1-1:nemo:services:schemas:data-certification-service:data-item	DataCertificationDataItem.xsd	Data Certification Service data item schema
dus:	urn:marlin:broadband:1-2:nemo:services:schemas:data-update-service	DataUpdate.xsd	Data Update Service schema
ls:	urn:marlin:broadband:1-1:nemo:services:schemas:license-service	License.xsd	License Service schema
mc:	urn:marlin:core:1-3:schemas	marlin-core.xsd	Marlin Core schema
mds:	urn:marlin:broadband:1-1:nemo:services:schemas:metering-service	Metering.xsd	Metering Data Service schema
rs:	urn:marlin:broadband:1-1:nemo:services:schemas:registration-service	Registration.xsd	Registration Service schema

In addition to the schemas defined by this specification, we leverage existing schemas to achieve our design goals. The following table summarizes the external schemas used in this specification:

Prefix	**XML Namespace**	**Description**
exc:	urn:marlin:core:1-2:nemo:services:schemas:exceptions	[MCS]
nemoc:	http://nemo.intertrust.com/2005/10/core	[NEMO] §3
nemosec:	http://nemo.intertrust.com/2005/10/security	[NEMO] §3
saml:	urn:oasis:names:tc:SAML:1.0:assertion	[SAML1.1]
wsse:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd	[WS-SEC]
wst:	http://schemas.xmlsoap.org/ws/2004/04/trust	[WS-TRUST]
xsd:	http://www.w3.org/2001/XMLSchema	[Schema]

1.4 Abbreviations

DCS	Data Certification Service
DCSA	Data Certification Standard Assertion

DUS	Data Update Service
HTTP	Hypertext Transfer Protocol
Marlin BB	Marlin Broadband
MDS	Metering Data Service
NEMO	Networked Environment for Media Orchestration
SAML	Security Assertions Markup Language
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
XML	Extensible Markup Language

1.5 Terms and Definitions

Please refer to the Terms and Definitions introduced in [MCS]. In addition, Marlin BB introduces the following:

Action Token	A Token that directs the client to perform a sequence of actions, such as obtaining a User Node from a Registration Service or acquiring a license from a License Service. An Action Token includes resource location information for the corresponding Configuration Token, and also information necessary to make protocol messages for communicating with the specified Marlin BB services.
Configuration Token	A Token that includes relatively static information for each of Marlin BB services (e.g., the service's WSDL definitions, node information, policy URIs, etc.).
License Suspension	Invalidation of certain Licenses, permanently or temporarily, disabling their use. License Suspensions are distributed in License Suspension Lists.
Service Token	Refers to an Action Token and its corresponding Configuration Token.
Store Web Site	An entity that is the front end for all the operations that interacts with the end user. As a result of such an interaction, a client is provisioned with a Configuration Token and an Action Token. Note that this entity is only used for illustration, as the same tokens could be delivered via another mechanism without affecting the specification.
Supported Octopus Nodes	Any of the following Octopus Node types; Personality, User (representing a domain) and Subscription. This specification enables various business models based on a node/link topology incorporating these Octopus Node types.

1.6 References

1.6.1 Normative References

[8pus]	Octopus DRM Technology Platform Specifications, Version 1.0
[MCS]	Marlin – Core System Specification, version1.3 and its latest errata
[NEMO]	NEMO Technology Platform Specifications, Version 1.1
[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.
[SAML1.1]	Eve Maler, Prateek Mishra and Rob Philpott, eds., *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1,* http://www.oasis-open.org/committees/download.php/3405/oasis-sstc-saml-bindings-1.1.pdf
[Schema]	XML Schema Part 1: Structures. W3C Recommendation. D. Beech, M. Maloney, N. Mendelsohn, H. Thompson. May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ XML Schema Part 2: Datatypes W3C Recommendation. P. Biron, A. Malhotra. May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
[SOAP11]	"Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman, Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C Note (08 May 2000). http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
[Starfish]	*Starfish - Marlin Broadcast Encryption Scheme* v1.2
[WS-SEC]	Web Services Security (WS-Security), Version 1.0, OASIS, April 5, 2002. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[WS-SECSAML]	Phillip Hallam-Baker *et al.,* eds., *Web Services Security: SAML Token Profile,* OASIS Standard, December 2004, http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
[WS-TRUST]	Web Services Trust Language (WS-Trust), Version 1.1, May 2004

1.6.2 Informative References

[SDMI]	The Secure Digital Music Initiative, July 8 1999, http://www.sdmi.org

2 Scope of Marlin BB

2.1 Overview (Informative)

There are currently no uniform requirements that content protection technologies must meet in order to be acceptable to Broadband Service Providers or Content Owners. While past initiatives (such as [SDMI]) have attempted to address this, there are few if no binding results to date, and current music, video or other content services implement a mix of usage and security features that are negotiated on a case-by-case basis between Content Owners, Service Providers, and Technology Providers.

Marlin BB specifies broadband-based client/service protocols, broadband service-managed domains, and DRM Client capabilities necessary to support most current content distribution business models.

Marlin BB is based on the same technologies as those defined in the Marlin Core Specification [MCS]. Marlin BB extends the capabilities of specific roles defined in [MCS], in cases where the extensions may be considered too onerous for all [MCS] implementations of the same role to support. However, it is conceivable that some of these extensions may migrate into [MCS] at a later stage.

In this current Marlin BB specification, domains are managed by a service, and licenses are targeted to users or DRM Clients registered with the service. However, it is easily conceivable that future versions of the specification may introduce the notion of locally managed domains and may define functionality by which Service Providers may issue licenses for targeting to these domains.

2.2 Relationship between Marlin BB and [MCS]

Marlin BB is an extension of [MCS]. For example,

- Marlin BB enhances the DRM Client role defined in [MCS] with additional capabilities.
- Marlin BB DRM objects adhere to the specifications defined in [MCS].

2.3 Specifications introduced in Marlin BB

Marlin BB extends the current [MCS]. Marlin BB introduces:

Broadband Domains
- Simple user-based domains where Octopus Personality Nodes are linked to User Nodes. Licenses may be targeted to any of the Supported Octopus Node types (i.e., User, Subscription or Personality Nodes)

DRM Client Functionality additional to that defined in [MCS]
- Support for license suspension and usage metering

Broadband-based Services
- Registering Users and DRM Clients
- Generating licenses

- Generating and certifying security-related metadata required for the execution of Marlin BB-specific usage rules
- Collection of metering data

Broadband-based Protocols

- Message exchanges between Marlin BB DRM Clients and Marlin BB Services. In this version of the specification, Marlin BB DRM Clients are the only clients that are expected to support these protocols. However, a later version of this specification will introduce additional client roles that support them (e.g., Broadband gateways used to enable communication between Marlin BB services and, for example, Marlin Core DRM Clients).

Marlin BB does not introduce new types of DRM objects. DRM objects produced or consumed by Marlin BB entities adhere to the specifications of a current or prior version of [MCS].

2.4 Marlin BB System Entities (Informative)

Marlin Entities are represented by the Marlin objects and roles (client or service functions) that realize the Marlin functional model.

A Marlin client or service is hosted by a NEMO Node that binds the client or service to a Marlin certified identity for authentication purposes, and provides it the keys necessary for message confidentially and integrity. Marlin assigns Roles to entities implementing the client and service functionalities. The Roles are certified by a Marlin Certification Authority and are necessary for establishing trust between clients and services.

Marlin Entities are to a large extent specified using Octopus and NEMO technologies (not redefined in this document). Marlin Core System Entities are defined in [MCS]. Marlin BB introduces additional Entities, defined in this section.

Marlin BB specifies an Internet-connected DRM Client and several broadband-based Services. Other Entities, such as content services and web stores for purchasing, are typically required for an end-to-end broadband content delivery and consumption system, but these do not intersect with DRM functionality.

Typically, a customer interacts with a Store Web Site to establish an account and to initiate registration of his or her devices and acquisition of licenses and content. The DRM Client interacts with the Registration Services, which are used to register clients and users, and with the License Service and the Content Service, which supply the DRM Client with License objects and protected content files, respectively. In addition, DRM Clients may interact with security metadata services (i.e., a Data Update Service or a Data Certification Service) that allow a service to deliver secure metadata to a client and ensure that a client uses a certain version of this security metadata. (An example of such security metadata is a License Suspension List.)

Note: Not mentioned are “personalization” services that may be deployed by a service to deliver NEMO and Octopus personalities as well as role assertions to Marlin BB clients and services. Such personalization may be accomplished during a first service interaction (for example, when DRM Client application software is distributed to general-purpose PC clients) or at manufacturing time, in the case of special-purpose devices.

2.4.1 Store Web Site

A Store Web Site is an optional entity that may be the front end for all the operations that interact with the end user. These operations may include the following:

- User (account) registration/deregistration
- Content acquisition (selection and payment).

As a result of the interactions with such an entity, the client is provisioned with Service Tokens that provide information necessary to communicate with Marlin BB services.

2.4.2 License Service

A License Service issues and delivers a License (composed of the appropriate Octopus objects) and may require submission of a Data Certification Standard Assertion (DCSA) proving that a defined set of security-related metadata has been legitimately acquired. When a service does not require a certain DCSA or security protocol for License provisioning, the service may choose to deliver Licenses in other ways (e.g., via email). Such alternative methods are beyond the scope of this specification. When such an alternative method cannot ensure interoperability among different types of client implementations, the standard protocol defined in this specification must also be supported by the service.

2.4.3 Registration Service

In Marlin BB, a Domain Manager is operated by a Service Provider, and a Domain, which is represented by a User, is managed via a Registration Service. This means that a Registration Service issues and manages the identities and the relationships (links between) the entities in the Domain, which can be DRM Clients, Marlin Users, and Subscriptions. When a service does not require a certain DCSA or security protocol for management of Link objects, the service may choose to deliver Link objects in other ways (e.g., via email). Such alternative methods are beyond the scope of this specification. When such an alternative method cannot ensure interoperability among different types of client implementations, the standard protocol defined in this specification must also be supported by the service.

2.4.4 Data Certification Service

A Marlin Data Certification Service (DCS) issues a proof (in the form of a DCSA) that assures certain security-related metadata items (e.g., secure clock or License Suspension Updates) at the DRM Client are reasonably up-to-date. A license can specify that release of a Content Key to a DRM Client is contingent on the DRM Client having such a proof. Certain service accesses can also be contingent on the DRM Client having such a proof.

2.4.5 Data Update Service

A Data Update Service (DUS) provides DRM Clients the latest security-related metadata items, such as trusted time and License Suspension Updates.

2.4.6 Metering Data Service

A Metering Data Service (MDS) receives metering data collected by the Marlin BB DRM Client, in accordance with the metering obligations expressed in the License.

2.4.7 Content Service

A Content Service is in charge of supplying clients with encrypted content files, typically by means of simple HTTP connections.

2.4.8 (Marlin BB) DRM Client

A Marlin BB DRM Client is a Marlin BB-compliant device that is able to directly communicate with Marlin BB services.

2.4.9 User

A (Marlin BB) User is the same as a Marlin User [MCS].

3 Marlin BB Domains

The rules that govern whether a Marlin BB DRM Client can join a domain, and what the consequences are when a DRM Client leaves a domain, are implemented by a Domain Manager and, to some extent, by the Marlin BB DRM Client, in accordance with the Domain Policy. In this first specification, a Domain Manager is implemented by a Registration Service.

3.1 Domain Policies in Marlin BB

Marlin BB leverages simple user-based domains, in which DRM Clients (Octopus Personality Nodes) are linked to Octopus User Nodes using Octopus Links. The creation and management of the User Nodes, as well as the generation of the Links, shall be implemented by a Registration Service. A single Registration Service may manage User Nodes and Links for a large number of Marlin Users.

3.1.1 Online Broadband Domain Policies

The Marlin BB Domain Policy is defined as follows:

- Octopus Personality Nodes are linked to an Octopus User Node.
- Registration Policy
    - Whether or not a DRM Client can be registered is determined by a Registration Service, based on data provided by the DRM Client in the registration request and on other information the service may already have.
- Deregistration Policy
    - If a DRM Client is deregistered, content whose license is targeted to the Domain from which the DRM Client deregistered MUST NOT be played on the deregistered DRM Client. Deregistration is specified in the deregistration protocol.
- User Links issued by a Registration Service include Domain ID and Domain Policy attributes, as specified in [MCS].
    - The Domain ID attribute value SHALL be the User Node ID
    - The Domain Policy attribute value SHALL be: urn:marlin:broadband:domain-policy:organization:<orgid>:*
        - <orgid> is the organization-specific identifier assigned from Marlin. Note this identifier can include suborganization identifiers which are managed by the organization itself.
        - * is any specific string within the given <orgid>.
    - An example Domain Policy attribute value is (Informative): urn:marlin:broadband:domain-policy:organization:acmeservice:policy:0

4 Marlin BB System Roles and Services

Marlin BB defines Roles and Services for a Marlin BB System.

4.1 Overview

This section defines the roles and services newly introduced in Marlin BB. A later section of this document specifies the messages and protocols by which clients and services communicate.

Roles	Services
License Service	License Service
Registration Service	Registration Service
Data Certification Service	Data Certification Service
Data Update Service	Data Update Service
Metering Data Service	Metering Data Service

The following table summarizes the set of URIs used as attribute values for conveying the above roles.

Role	URI
License Service	urn:marlin:broadband:role:license-service
Registration Service	urn:marlin:broadband:role:registration-service
Data Certification Service	urn:marlin:broadband:role:data-certification-service
Data Update Service	urn:marlin:broadband:role:data-update-service
Metering Data Service	urn:marlin:broadband:role:metering-data-service

In a Marlin BB system, a client for Marlin BB services is REQUIRED to have a DRM Client role which is defined in [MCS] §4 with additional capabilities to communicate with Marlin BB services.

4.2 Roles Definitions

Note: According to [NEMO] §4, roles shall be encoded as SAML 1.1 attribute assertions.

4.2.1 License Service

Each NEMO Node implementing this role shall issue Licenses (composed of the appropriate Octopus objects) to clients. The trust authority of the role assertion for the License Service SHALL be the DRM Services Authority.

4.2.2 Registration Service

Each NEMO Node implementing this role issues User Nodes, Subscription Nodes, User Links, and Subscription Links to clients. In addition, this service handles deregistration of Supported Octopus Nodes. The trust authority of the role assertion for the Registration Service SHALL be the DRM Services Authority.

4.2.3 Data Certification Service

Each NEMO Node implementing this role shall issue assertions that certify that certain client security-related metadata (e.g., secure clock or License Suspension Updates) are reasonably up-to-date. Such assertions may be required by Controls in a license, or by Marlin BB services, before they will accept clients' requests. The trust authority of the role assertion for the Data Certification Service SHALL be the Data Certification Services Authority.

4.2.4 Data Update Service

Each NEMO Node implementing this role shall issue security-related metadata (e.g., secure clock or License Suspension Updates) to DRM Clients. The trust authority of the role assertion for the Data Update Service SHALL be the Data Certification Services Authority.

4.2.5 Metering Data Service

Each NEMO Node implementing this role shall be certified to receive metering data collected by Marlin DRM Clients, in accordance with the metering obligations expressed in Licenses. The trust authority of the role assertion for the Metering Data Service SHALL be the DRM Services Authority.

5 Marlin BB System Protocols

Marlin BB System Protocols SHALL use the NEMO SOAP/HTTP Message Bindings.

5.1 Message Security

In Marlin BB, in order to accommodate a Marlin BB service (such as DUS) or a newly introduced assertion (DCSA), additional message security specifications beyond those in [MCS] are defined.

5.1.1 Message Security Policy

The following Protocol Policy is defined in addition to [MCS] §5.2. Since Freshness is not required for the Protocol Policy, Timestamp is OPTIONAL.

Protocol Policy	Integrity	Nonce	Timestamp	Confidentiality
Integrity + Confidentiality	YES	YES	OPTIONAL	YES

5.1.2 Message Faults

This specification defines extensions of message faults defined in [MCS] §5.3 in the context of Marlin BB. A Marlin BB-compliant implementation MUST implement both the behaviours defined in [MCS] and the ones defined in this specification.

When there is a SOAP [SOAP11] processing fault (e.g. if the server faults while processing HTTP headers related to the SOAP binding (e.g. SOAPAction), or if the server faults while processing SOAP elements or attributes), the server SHALL return an HTTP 500 and SHALL, either:

- NOT supply a <detail> element, or,
- Supply an EMPTY <detail> element

in the body of the fault.
When HTTP 500 is returned, the client is free to process such an error any way it wants.

5.1.2.1 Faults for SOAP Header Processing

When there is no SOAP processing fault, the server SHALL return a HTTP 200 in all cases. For SOAP header processing errors, contents in SOAP Envelope are specified as following:

- When the request processing identifies one of the errors described in this section, then the responder SHALL return a soap fault response as described in this section. In these cases, the soap fault message SHALL use message security policy as defined in [MCS] §12.3.3.
- All cases other than those described this section, the responder SHALL return a fault message as depicted below:

```
<SOAP-ENV:Envelope
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:xsd="http://www.w3.org/1999/XMLSchema"
         xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Body>
```

```
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>Unspecified Error</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following string identifiers are defined for the name attribute of the <exc:ServiceException> element, and associated behaviors for the <exc:Details> element.

exc:ServiceException/@name

The following string identifiers are defined for this attribute:

- bbexc:DCSAssertionStaleException: This indicates that a DCSA in the request message is older than the service can accept.
- bbexc:DCSAssertionMissingException: This indicates that a DCSA required by the service is not found in the request message.
- bbexc:BroadbandVersionUnacceptableException: This indicates that the version of the Marlin BB specification supported by the client (as indicated by the client role assertion) is lower than the minimum specification version required by the service.
- bbexc:ClientObsoleteException: This indicates that the client (the organization-specific version of the client, as specified in the client role assertion) is one identified as being hacked, so all services have been directed to shun it (refuse to provide services to it).

exc:ServiceException/exc:Details

The following behaviors for the <exc:Details> element are defined in Marlin BB:

- For exc:ServiceException/@name="bbexc:DCSAssertionStaleException": The <exc:Details> element MUST be present and MUST contain the data certification standard name of the DCSA that is stale. The client SHOULD acquire a new DCSA for the specified data certification standard name and retry the operation.
- For exc:ServiceException/@name="bbexc:DCSAssertionMissingException": The <exc:Details> element MUST be present and MUST contain the data certification standard name of the DCSA that is missing. The client SHOULD acquire a DCSA for the specified data certification standard name and retry the operation.
- For exc:ServiceException/@name="bbexc:BroadbandVersionUnacceptableException": The <exc:Details> element SHALL NOT be present. The user should be directed to upgrade to the latest version of the client.
- For exc:ServiceException/@name="bbexc:ClientObsoleteException": The <exc:Details> element SHALL NOT be present. The user should be directed to upgrade to the latest version of the client.

5.1.2.2 Faults for SOAP Body Processing

No additional string identifiers for the name attribute of the <exc:ServiceException> element are defined for Marlin BB in this context.

5.1.3 Inclusion of a DCSA in a Request Message

A Data Certification Standard Assertion (DCSA), when required in a request message by a certain service, SHALL be placed as a direct child element of a <wsse:Security> element, and SHALL be referenced from a <wsse:KeyIdentifier> element in a <wsse:SecurityTokenReference> element, as specified in [WS-SECSAML] §3.3.
The client MAY also include DCSA(s) which is not required by a certain service in a request message. The service is only REQUIRED to check the necessary DCSA(s) in a request message, and when the message includes other DCSA(s), the service is REQUIRED to ignore other DCSA(s).
For a certain data certification standard name, a client is REQUIRED to include only the latest DCSA possessed by the client in a request message. In other words, for a certain data certification standard name, a client SHALL NOT include more than one DCSA in a request message.

The <wsse:SecurityTokenReference> element that references a SAML attribute assertion that asserts a DCSA SHALL contain a nemosec:Usage attribute with the value

```
urn:marlin:broadband:1.0:nemo:services:datacertification-service:assertion
```

5.1.4 Clock Synchronization

Each of the Marlin BB services is required to maintain accurate time. When a Timestamp (UTC) is required in a response message, the service supplying the message shall use this time as the source of the Timestamp. DRM Clients that support trusted time should update their clocks based on Timestamps in response messages. However DRM Clients SHOULD NOT update their clocks based on Timestamps in fault response messages.

5.2 Service-specific Protocols

5.2.1 License Service

5.2.1.1 Overview

A License Service handles the creation of License objects, which govern access to protected content. License objects are targeted to Octopus Nodes.

Copies of the XML schema and the WSDL for a License Service are in Appendices A.1 and B.1, respectively.

5.2.1.2 Request Parameters

- *<oct:Bundle>:* a data structure containing the public part of the Octopus Node to which the License is to be bound. Client implementations SHALL be able to deliver the public part of either a Personality Node or a User Node for this parameter. Client implementations MAY supply the public part of other Supported Octopus Node types.
- *<ls:BusinessToken>:* an opaque data structure containing service-specific data retrieved from the Action Token (see §5.3).

5.2.1.3 Response Data

- *<oct:Bundle>:* a data structure containing the License, which consists of Control, Controller, Content Key(s), and Protector objects related to the Octopus Node provided in the request. The License Service MAY include one or more context IDs, as described in [MCS] §3.2.4

5.2.1.4 Protocol for the License Service Security Policy

The request MUST obey the 'Full Security' policy defined in [MCS] §5.2.

In addition, the client's Role assertion is attached to the request, and appropriate DCSA(s) are also attached to the request if data certification standard name(s) are specified in the corresponding action in the Action Token. (See §5.3.2.)

The response MUST obey the 'Full Security' policy defined in [MCS] §5.2.

In order to correlate the request with the response message, the Message Correlation pattern described in [NEMO] §2.3 MUST be used. The specific information in the SOAP header guaranteeing the correlation MUST be covered by the message signature.

The identifier for the License Service policy is:

```
urn:marlin:broadband:1.0:nemo:services:license-service:policy:0
```

5.2.2 Registration Service

5.2.2.1 Overview

The Registration Service provides the following three operations:

- Issuance of User and Subscription Nodes
- Issuance of User and Subscription Links
- Deregistration of Supported Octopus Nodes.

Copies of the XML schema and the WSDL for a Registration Service are in Appendices A.2 and B.2, respectively.

5.2.2.2 Node Acquisition

Node Acquisition is a simple REQUEST/RESPONSE protocol used to obtain a User Node or a Subscription Node.

5.2.2.2.1 Request Parameters

- *<rs:BusinessToken>:* an opaque data structure containing service-specific data retrieved from the Action Token.

5.2.2.2.2 Response Data

- *<oct:Bundle>:* a data structure containing the public part of the acquired Node (User or Subscription).

5.2.2.2.3 Protocol for the Registration Service Node Acquisition Security Policy

The request MUST obey the 'Full Security' policy defined in [MCS] §5.2.

In addition, the client's Role assertion is attached to the request, and appropriate DCSA(s) are also attached to the request if one or more data certification standard names are specified in the corresponding action in the Action Token. (See §5.3.2.)

The response MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

In order to correlate the request message with the response message, the Message Correlation pattern described in [NEMO] §2.3 MUST be used. The specific information in the SOAP header guaranteeing the correlation MUST be covered by the message signature.

The identifier for the Registration Service Node Acquisition policy is:

```
urn:marlin:broadband:1.0:nemo:services:registration-service:nodeAcquisition:policy:0
```

5.2.2.3 Link Acquisition

Link Acquisition is a REQUEST/RESPONSE/CONFIRMATION protocol used to establish a relationship (i.e., obtain an Octopus Link, usually a User Link or a Subscription Link) between Octopus Nodes. The exchange allows for the response to bear an Agent (see [8pus] §3). The Agent is obligated to supply, in a confirmation message, the result of processing the Agent. Confirming to the Registration Service allows the service to determine that the Agent has been processed in a secure environment.

5.2.2.3.1 Request Parameters

- *<oct:Bundle>:* a data structure that SHALL contain the public part of an Octopus Node that is to be referenced by the <oct:LinkFrom> element in the acquired Link Object. Client implementations SHALL be able to supply the public part of either a Personality Node or a User Node for this parameter. Client implementations MAY supply the public part of other Supported Octopus Nodes.
- *<oct:Bundle>:* a data structure that SHALL contain the public part of an Octopus Node that is to be referenced by the <oct:LinkTo> element in the acquired Link Object. Client implementations SHALL be able to supply the public part of either a User Node or a Subscription Node for this parameter. Client implementations MAY supply the public part of other Supported Octopus Nodes.
- *<rs:BusinessToken>:* an opaque data structure containing service-specific data retrieved from the Action Token.

5.2.2.3.2 Response Data

- *<oct:Bundle>:* a data structure containing the acquired Link Object.
- *<mc:AgentCarrier>:* an OPTIONAL element that supplies the data structure containing the Agent, the input parameters, and the context ID.

5.2.2.3.3 Confirmation parameters

- *<mc:AgentResultBlock>:* an OPTIONAL element that represents the data resulting from processing the Agent. When the response does not contain an <AgentCarrier> element, the <mc:AgentResultBlock> element SHALL be omitted in the confirmation.

5.2.2.3.4 Protocol for the Registration Service Link Acquisition Security Policy

The request MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

In addition, the client's Role assertion is attached to the request, and DCSA(s) are also attached to the request if one or more data certification standard names are specified in the corresponding action in the Action Token. (See §5.3.2.)

The response MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

The confirmation MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

In order to correlate the request message with the response or confirmation messages, the Message Correlation pattern described in [NEMO] §2.3 MUST be used. The specific information in the SOAP header guaranteeing the correlation MUST be covered by the message signature.

The identifier for the Registration Service Link Acquisition policy is:

```
urn:marlin:broadband:1.0:nemo:services:registration-service:linkAcquisition:policy:0
```

5.2.2.4 Deregistration

Deregistration is a REQUEST/RESPONSE/CONFIRMATION protocol used to terminate a relationship (the result of the Link Acquisition protocol) between Octopus Nodes (e.g., between a Personality Node and a User Node or between a User Node and a Subscription Node). The response bears an Agent (see [8pus] §3). The Agent is obligated to supply, in a confirmation message, the result of processing the Agent. Confirming to the Registration Service allows the service to determine that the Agent has been processed in a secure environment.

5.2.2.4.1 Request Parameters

- *<oct:Bundle>:* a data structure containing the public part of an Octopus Node. Client implementations SHALL be able to supply the public part of a Personality Node for this parameter. Client implementations MAY support supplying the public part of other Supported Octopus Nodes.
- *<oct:Bundle>:* a data structure containing the public part of an Octopus Node. Client implementations SHALL be able to supply a User Node for this parameter. Client implementations MAY supply the public part of other Supported Octopus Nodes.
- *<rs:BusinessToken>:* an opaque data structure containing service-specific data received from the Action Token.

5.2.2.4.2 Response Data

- *<mc:AgentCarrier>:* a data structure containing the Agent, the input parameters, and the context ID.

5.2.2.4.3 Confirmation parameters

- *<mc:AgentResultBlock>:* a data structure containing the result of processing the Agent.

5.2.2.4.4 Protocol for the Registration Service Deregistration Security Policy

The request MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

In addition, the client’s Role assertion is attached to the request.

The response MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

The confirmation MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

In order to correlate the request message with the response or confirmation messages, the Message Correlation pattern described in [NEMO] §2.3 MUST be used. The specific information in the SOAP header guaranteeing the correlation MUST be covered by the message signature.

The identifier for the Registration Service Deregistration policy is:

urn:marlin:broadband:1.0:nemo:services:registration-service:deregistration:policy:0

5.2.3 Data Certification Service

5.2.3.1 Overview

A Marlin Data Certification Service(DCS) is used to determine whether the current security-related metadata of the client is up-to-date.

A Service Provider MAY define a data certification standard to force clients to acquire certain security-related metadata as a prerequisite to interaction with certain Marlin BB services (i.e., License Services and Registration Services). A data certification standard represents a set of security-related metadata attributes and their values. These values MAY be time-dependent. A Data Certification Standard Assertion (DCSA) represents the assertion that a particular principal’s security-related metadata values were up-to-date with current values at the time the assertion was acquired.

When a data certification standard is required by a certain service, as specified in an Action Token, the client is required to provide a DCSA to prove to the service that its security-related metadata is sufficiently up-to-date. The more time has passed since the assertion was issued, the higher the risk that the client no longer meets the data certification standard.

Therefore, a Marlin BB service MAY implement a policy that defines how old the assertion can be before it becomes unacceptable. For strong assurance, the service MAY require that the assertion be no more than a few minutes old, essentially requiring the client to acquire the assertion immediately before interacting with the service.

A service MAY require the client to meet multiple distinct data certification standards. A distinct DCSA is required for each of the data certification standards.

Marlin BB services MAY also require a client to meet a data certification standard in order to consume some or all types of content. This is achieved by encoding the requirement for the presence of a DCSA for one or more data certification standards in the License for the relevant pieces of content. Note that this constraint MAY only be applied for clients whose device-class is in the class(es) of concern. (If consumption is allowed on other types of devices, the License MUST condition the constraint based on the device-class.)

When a client is refused by a Marlin BB service or it is unable to successfully evaluate a License because it is missing or has one or more outdated DCSAs, it SHOULD interact with the appropriate DCS(s) to acquire current DCSAs, based on the information specified in the Configuration Token. In each DCS request, the client SHOULD provide the names (i.e., namespaces and names) and values of security-related metadata items that are related to the intended data certification standard(s). When certain security-related metadata required for the data certification standard is not yet issued from the DUS (e.g., if no License Suspension Update has been issued), then, in the DCS request, the client SHOULD only send the names and values of the security-related metadata items the client could acquire at the time. In other words, the client SHOULD NOT send names of security-related metadata with empty values for not-yet-acquired security-related metadata items.

A DCS MAY, for the data certification standards it is authorized to certify, apply the data certification standard policies for the particular client type to determine whether it was provided all necessary information and that information is acceptable. When a DCS determines that the provided security-related metadata items satisfy the conditions for the requested data certification standard(s), the DCS responds with one or more DCSAs, one per each requested standard. When the metadata items do not meet the requirements for the data certification standard(s) , the DCS responds with the names (namespaces and names) of the security-related metadata that need to be updated.

Copies of the XML schema and the WSDL for a Data Certification Service are in Appendices A.3 and B.3, respectively.

The XML schema for the <dcsi:DataItem> and <dcsi:DataItemSet> elements is in Appendix A.8.

5.2.3.2 Request Parameter

- *<wst:RequestSecurityToken>:* a data structure containing information identifying the security-related metadata for a specific data certification standard. The request contains one <RequestSecurityToken> per data certification standard for which a DCSA is needed.

The syntax for a <wst:RequestSecurityToken> element is as follows:

wst:RequestSecurityToken
    The child element of the DCS request payload.
wst:RequestSecurityToken/wst:TokenType
    The <wst:TokenType> element SHALL contain the following URI:
    `urn:marlin:broadband:1.0:nemo:services:datacertification-service:assertion`
wst:RequestSecurityToken/wst:Claims/dcsi:DataItemSet
    The <dcsi:DataItemSet> element specifies a data certification standard for which a DCSA is requested.
.../dcsi:DataItemSet/@name
    The name attribute SHALL contain the following value:
    `certification-standard-name`

.../dcsi:DataItemSet/@namespace

The namespace attribute SHALL contain the following value:

urn:marlin:broadband

.../dcsi:DataItemSet/@value

The value attribute SHALL contain the certification standard name, which is uniquely defined by a URI.

wst:RequestSecurityToken/wst:Supporting/dcsi:DataItem

The <dcsi:DataItem> element specifies a data item to be validated. There is one <dcsi:DataItem> per each metadata item required by the data certification standard, except in the following situation: When there is no corresponding value for the data item in the client, a <dcsi:DataItem> element for the data item SHALL NOT appear in the <wst:Supporting> element..

.../dcsi:DataItem/@name

The name attribute SHALL contain the name of the data item.

.../dcsi:DataItem/@namespace

The namespace attribute SHALL contain the namespace of the data item.

.../dcsi:DataItem/dcsi:Value

The <dcsi:Value> element SHALL contain the value of the data item to be validated.

Elements or attributes specified other than here SHALL NOT be used in the <wst:RequestSecurityToken> element.

The following is a set of namespaces and names that can be used as attributes in the <dcsi:DataItem> element of the <wst:RequestSecurityToken> element.

Namespace	Name	Value Type
urn:marlin:broadband:security-metadata:attributes	trusted-time	dateTime
urn:marlin:broadband:security-metadata:attributes:license-suspension-serial	(specific service name)	nonNegativeInteger
urn:marlin:broadband:security-metadata:attributes	crlNumber	nonNegativeInteger
urn:marlin:broadband:security-metadata:attributes	bkbRevocationVersion	nonNegativeInteger

Note: The value of the bkbRevocationVersion attribute is the Revocation Version, which is specified in [Starfish] §4.1.

Below is an example of a <dcs:DataCertificationAcquisitionRequestPayload> element. In this example, the data certification standard name value is specified as "urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME", where “CERTIFICATION_STANDARD_NAME” is a uniquely defined value within the acmeorganization URI. This example assumes that the specified data certification standard requires the following security-related metadata information:

- trusted-time
- license-suspension-serial for urn:marlin:organization:acme
- crlNumber
- bkbRevocationVersion

When no CRL has been issued yet (from CRL Distribution Points), the client only includes the following security-related metadata information, as in the example:

- trusted-time
- license-suspension-serial for urn:marlin:organization:acme
- bkbRevocationVersion

```
<DataCertificationAcquisitionRequestPayload xmlns="urn:marlin:broadband:1-
1:nemo:services:schemas:data-certification-service">
  <wst:RequestSecurityToken>
    <wst:TokenType>urn:marlin:broadband:1.0:nemo:services:datacertification-
service:assertion</wst:TokenType>
    <wst:Claims>
      <dcsi:DataItemSet namespace="urn:marlin:broadband" name="certification-standard-
name" value="urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME"/>
    </wst:Claims>
    <wst:Supporting>
      <dcsi:DataItem name="trusted-time" namespace="urn:marlin:broadband:security-
metadata:attributes">
        <dcsi:Value xsi:type="xsd:dateTime">2006-09-20T14:30:27Z</dcsi:Value>
      </dcsi:DataItem>
      <dcsi:DataItem name="urn:marlin:organization:acme"
namespace="urn:marlin:broadband:security-metadata:attributes:license-suspension-serial">
        <dcsi:Value xsi:type="xsd:nonNegativeInteger">32</dcsi:Value>
      </dcsi:DataItem>
      <dcsi:DataItem name="bkbRevocationVersion"
namespace="urn:marlin:broadband:security-metadata:attributes">
        <dcsi:Value xsi:type="xsd:nonNegativeInteger">1</dcsi:Value>
      </dcsi:DataItem>
    </wst:Supporting>
  </wst:RequestSecurityToken>
</DataCertificationAcquisitionRequestPayload>
```

5.2.3.3 Response Data

- *<wst:RequestSecurityTokenResponse>:* a data structure containing a <saml:Assertion> element (i.e., a DCSA) if the request for the DCSA is granted. If the request is not granted, the set of namespaces and names defined in §5.2.3.2 that must be updated is returned in the <dcsi:DataItemSet> element of the <RequestSecurityTokenResponse>. The DCS response contains one <RequestSecurityTokenResponse> per DCSA requested.

The syntax for a <wst:RequestSecurityTokenResponse> element is as follows:

wst:RequestSecurityTokenResponse
    The child element of the DCS response payload.
wst:RequestSecurityTokenResponse/wst:RequestedSecurityToken
    The <wst:RequestedSecurityToken> element contains the requested Data Certificate.
.../wst:RequestedSecurityToken/saml:Assertion
    This element appears if the DCSA is to be returned as a SAML attribute assertion, conforming to [SAML1.1].

When the security-related metadata submitted in the DCS request does not meet the requirements for the data certification standard, the DCS responds with the names (namespaces and names) of the security-related metadata that must be updated. If a value for a security-related metadata item was sent in the request message, the value is also returned in the response. The syntax for the elements and attributes returned in the response when a request is not granted is as follows:

wst:RequestSecurityTokenResponse/wst:TokenType

The <wst:TokenType> element SHALL contain the following URI:

`urn:marlin:broadband:1.0:nemo:services:datacertification-service:assertion`

wst:RequestSecurityTokenResponse/wst:Claims/dcsi:DataItemSet

The <dcsi:DataItemSet> element specifies a data certification standard that was requested.

.../dcsi:DataItemSet/@name

The name attribute SHALL contain the following value:

`certification-standard-name`

.../dcsi:DataItemSet/@namespace

The namespace attribute SHALL contain the following value:

`urn:marlin:broadband`

.../dcsi:DataItemSet/@value

The value attribute SHALL contain a data certification standard name, which is uniquely defined by a URI.

wst:RequestSecurityTokenResponse/wst:Supporting/dcsi:DataItem

Each <dcsi:DataItem> element contains a data item that SHOULD be updated to meet the requirements for the data certification standard specified by the <dcsi:DataItemSet> value.

.../dcsi:DataItem/@name

The name attribute SHALL contain the name of the data item.

.../dcsi:DataItem/@namespace

The namespace attribute SHALL contain the namespace of the data item.

.../dcsi:DataItem/dcsi:Value

The optional <dcsi:Value> element SHALL contain the value of the data, when the corresponding value was sent in the request.

Elements or attributes specified other than here SHALL NOT be used in the <wst:RequestSecurityTokenResponse> element.

Below is an example of a <dcs:DataCertificationAcquisitionResponsePayload> element. In this example, the data certification standard name value is specified as "urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME", where “CERTIFICATION_STANDARD_NAME” is a uniquely defined value within the acme URI. The example assumes that the specified data certification standard requires the following security-related metadata information:

- trusted-time
- license-suspension-serial for urn:marlin:organization:acme
- crlNumber
- bkbRevocationVersion

The example assumes that in the DCS request, the client did not send a license-suspension-serial value, and the crlNumber sent in the request does not meet the requirements for the data certification standard. As a result, the following security-related metadata information items are returned:

- license-suspension-serial (urn:marlin:organization:acme) without a value
- crlNumber with the value that was sent in the request

```
<DataCertificationAcquisitionResponsePayload xmlns="urn:marlin:broadband:1-
1:nemo:services:schemas:data-certification-service">
  <wst:RequestSecurityTokenResponse>
    <wst:TokenType>urn:marlin:broadband:1.0:nemo:services:datacertification-
service:assertion</wst:TokenType>
    <wst:Claims>
      <dcsi:DataItemSet namespace="urn:marlin:broadband" name="certification-standard-
name" value="urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME"/>
    </wst:Claims>
    <wst:Supporting>
      <dcsi:DataItem name="urn:marlin:organization:acme"
namespace="urn:marlin:broadband:security-metadata:attributes:license-suspension-serial"/>
      </dcsi:DataItem>
      <dcsi:DataItem name="crlNumber" namespace="urn:marlin:broadband:security-
metadata:attributes">
        <dcsi:Value xsi:type="xsd:nonNegativeInteger">1</dcsi:Value>
      </dcsi:DataItem>
    </wst:Supporting>
  </wst:RequestSecurityTokenResponse>
</DataCertificationAcquisitionResponsePayload>
```

5.2.3.4 Protocol for the Data Certification Service Security Policy

The request MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.
In addition, the client’s Role assertion is attached to the request.

The response MUST obey the ‘Full Security’ policy defined in [MCS] §5.2.

In order to correlate the request message with the response message, the Message Correlation pattern described in [NEMO] §2.3 MUST be used. The specific information in the SOAP header guaranteeing the correlation MUST be covered by the message signature.

The identifier for Data Certification Service policy is:

```
urn:marlin:broadband:1.0:nemo:services:datacertification-service:policy:0
```

5.2.3.5 Data Certification Standard Assertion

A DCSA asserts that the client met the requirements of a specific data certification standard at the time the assertion was issued.

An attribute for a DCSA is defined as follows:

Namespace	Name	Object Path

Namespace	Name	Object Path
urn:marlin:broadband	certification-standard-name	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:certification-standard-name

The value of the attribute is an organization-specific value under "urn:marlin:organization" that names the certification standard.

The authorization takes the form of a SAML attribute assertion. Here is an example:

```
  <saml:AttributeStatement>
     <saml:Subject>
        <saml:NameIdentifier Format="http://nemo.intertrust.com/2004/NEMONodeID">urn:organization-
identifier:nemo-personality:0000000000000001</saml:NameIdentifier>
     </saml:Subject>
     <saml:Attribute AttributeNamespace="urn:marlin:broadband:" AttributeName="certification-standard-
name">
        <saml:AttributeValue>urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME</saml:
AttributeValue>
     </saml:Attribute>
  </saml:AttributeStatement>
```

5.2.4 Data Update Service

5.2.4.1 Overview

A Marlin Data Update Service (DUS) provides clients current security-related metadata items, such as trusted-time and License Suspension Update.

The client provides the DUS the list of names and (optionally) values of security-related metadata for which updates are being requested. The DUS returns metada when the service has newer information than the data provided from the client.

When the client is unable to acquire/update a DCSA from a DCS, the client SHOULD interact with a DUS to acquire current security-related metadata. Once the client has been updated, it SHOULD reattempt an acquisition of the requisite DCSA(s) from a DCS.

Copies of the XML schema and the WSDL for a Data Update Service are in Appendices A.4 and B.4, respectively.

The XML schema for the <dcsi:DataItem> element is in Appendix A.8.

5.2.4.2 Request Parameter

- *<dcsi:DataItemSet>:* a data structure containing <dcsi:DataItem>(s) that identify security-related metadata. The same set of namespaces and names defined in § 5.2.3.2 are used as attributes in the <dcsi:DataItem> elements of <dcsi:DataItemSet>.

The syntax for the <dcsi:DataItemSet> element is as follows:

dcsi:DataItemSet

The child element of the DUS request payload.

dcsi:DataItemSet/@name

The name attribute SHALL contain the following value:

attributes

dcsi:DataItemSet/@namespace

The namespace attribute SHALL contain the following value:

urn:marlin:broadband:security-metadata

dcsi:DataItemSet/dcsi:DataItem

Each <dcsi:DataItem> element specifies a data item to be updated.

.../dcsi:DataItem/@name

The name attribute SHALL contain the name of the data item.

.../dcsi:DataItem/@namespace

The namespace attribute SHALL contain the namespace of the data item.

.../dcsi:DataItem/dcsi:Value

The <dcsi:Value> element SHALL contain the value of the data item to be updated , when there is a corresponding value in the client. Otherwise, the <dcsi:Value> element SHALL be omitted.

Elements or attributes specified other than here SHALL NOT be used in the <dcsi:DataItemSet> element.

Below is an example of a <dus:DataUpdateRequestPayload> element. In this example, the client requests the following security-related metadata:

- trusted-time
- License Suspension Update for urn:marlin:organization:acme

The client includes values for the following security-related metadata, because the client has values for that metadata:

- trusted-time
- License Suspension Update for urn:marlin:organization:acme

Since the client does not have values for the other metadata items, a <dcsi:Value> element is not included in the <dcsi:DataItem> elements for such metadata.

```
<DataUpdateRequestPayload xmlns="urn:marlin:broadband:1-
2:nemo:services:schemas:data-update-service">
  <dcsi:DataItemSet namespace="urn:marlin:broadband:security-metadata"
name="attributes">
    <dcsi:DataItem name="trusted-time" namespace="urn:marlin:broadband:security-
metadata:attributes">
      <dcsi:Value xsi:type="xsd:dateTime">2006-09-20T14:30:27Z</dcsi:Value>
    </dcsi:DataItem>
    <dcsi:DataItem name="urn:marlin:organization:acme"
namespace="urn:marlin:broadband:security-metadata:attributes:license-suspension-
serial">
      <dcsi:Value xsi:type="xsd:nonNegativeInteger">32</dcsi:Value>
    </dcsi:DataItem>
  </dcsi:DataItemSet>
</DataUpdateRequestPayload>
```

5.2.4.3 Response Data

- *<dus:DataUpdateSet>:* a data structure containing the metadata that is updated by the DUS. TrustedTimeUpdate and LicenseSuspensionUpdate(s) can be provided from the DUS.

5.2.4.4 Protocol for the Data Update Service Security Policy

The request MUST obey the 'Integrity + Confidentiality' policy defined in §5.1.1. In addition, the client's Role assertion is attached to the request.

The response MUST obey the 'Full Security' policy defined in [MCS] §5.2.

In order to correlate the request message with the response message, the Message Correlation pattern described in [NEMO] §2.3 MUST be used. The specific information in the SOAP header guaranteeing mostly the correlation MUST be covered by the message signature.

The identifier for Data Update Service policy is:

```
urn:marlin:broadband:1.0:nemo:services:dataupdate-service:policy:0
```

5.2.5 Metering Data Service

5.2.5.1 Overview

The Metering Data Service (MDS) receives a client's metering information. The support for MDS is REQUIRED only for clients which have meter-play duration capability described in § 7.1.2.

Copies of the XML schema and the WSDL for a Metering Data Service are in Appendices A.5 and B.5, respectively.

5.2.5.2 Request Parameter

- *<mds:MeteringData>:* a data structure containing recorded metering information, if it is available, or, if it is not, the reason is not. The details of the <MeteringData> element are described in §6.3.

5.2.5.3 Response Data

When the service successfully receives the request messages, it returns an empty payload.

5.2.5.4 Protocol for the Metering Data Service Security Policy

The request MUST obey the 'Full Security' policy defined in [MCS] §5.2. In addition, the client's Role assertion is attached to the request.

The response MUST obey the 'Integrity + Freshness' policy defined in [MCS] §5.2.

In order to correlate the request message with the response message, the Message Correlation pattern described in [NEMO] §2.3 MUST be used. The specific information in the SOAP header guaranteeing the correlation MUST be covered by the message signature.

The identifier for Metering Data Service policy is:

```
urn:marlin:broadband:1.0:nemo:services:metering-service:policy:0
```

5.3 Service Tokens

Broadband service interactions typically require some form of user login to authorize actions such as device registration or license acquisition. The result of this user-based transaction can be a secret token to be used in further request processing. It is beyond the scope of this document to define standards for user authentication and resulting secret token content data. However, since Marlin BB service protocols require that such secret data be included in request messages, tokens are specified that wrap and identify the secret data, and Marlin BB services MUST issue these tokens.

Two tokens are specified in this document: Configuration Token, and Action Token.

5.3.1 Configuration Token

A Configuration Token, which is expressed by a <bsc:BroadbandServiceConfig> element, can contain one or more of respective service configurations. When a <BroadbandServiceConfig> element contains multiple License Service Configurations or Registration Service Configurations, the id attribute SHALL be specified in each of those service configurations. The values specified by the id attribute SHALL be unique in the <BroadbandServiceConfig> element when they are specified.
The BroadbandServiceConfig element has three mandatory attributes:

- broadbandServiceId: Identifies uniquely the broadband service that uses the services described in this configuration.
- configVersion: the version of this configuration. This value can only increase over time.
- broadbandServiceFriendlyName: a human readable name for this broadband service.

A copy of the XML schema for a Configuration Token is in Appendix A.6.

5.3.1.1 License Service Configuration

A <bsc:LicenseServiceConfig> element corresponds to a License Service. This element MUST include the following information:

- <nemoc:NodeInfo>: signing, encryption NEMO keys, and role assertion information for the License Service.
- <wsdl:definitions>: WSDL definitions for the License Service.
- <bsc:PolicyURI>: the Policy URI which is applied to the operation of license service wsdl. The operation is identified by.operationName attribute. As the Policy URI, the License Service policy defined in §5.2.1.4 SHALL be specified.

Here is an example of a <LicenseServiceConfig> element:

```
<LicenseServiceConfig xmlns="urn:marlin:broadband:1-2:nemo:services:configuration"
```

```
id="1">
  <nemoc:NodeInfo>
    <!-- License Service Node's Encryption Key -->
    <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-encryptionKey">
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
     <!-- License Service Node's Signing Key -->
     <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-signingKey">
        <wsse:Embedded>…..</wsse:Embedded>
      </wsse:SecurityTokenReference>
      <!-- License Service’s Role Assertion -->
      <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2004/attribute/role">
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
  </nemoc:NodeInfo>
  <!-- License Service’s WSDL -->
  <wsdl:definitions>…</wsdl:definitions>
  <!-- License Service’s Policy-->
  <PolicyURI operationName="
requestLicense">urn:marlin:broadband:1.0:nemo:services:license-
service:policy:0</PolicyURI>
</LicenseServiceConfig>
```

5.3.1.2 Registration Service Configuration

A <bsc:RegistrationServiceConfig> element corresponds to a Registration Service. This element MUST include the following information:

- <NodeInfo>: signing, encryption NEMO keys, and role assertion information for the Registration Service.
- <wsdl:definitions>: WSDL definitions for the Registration Service.
- <bsc:PolicyURI>: the Policy URIs which are applied to the operations of registration service wsdl. Each of operations is identified by.operationName attribute. For registration and deregistration, the specified Policy URI is also applied to the operation for confirmation message.
    - For nodeAcquisition, the Registration Service Node Acquisition policy defined in §5.2.2.2.3 SHALL be specified as the Policy URI.
    - For linkAcquisition, the Registration Service Link Acquisition policy defined in §5.2.2.3.4 SHALL be specified as the Policy URI.
    - For deregistration, the Registration Service Deregistration policy defined in §5.2.2.4.4 SHALL be specified as the Policy URI.

Here is an example of a <RegistrationServiceConfig> element:

```
<RegistrationServiceConfig xmlns="urn:marlin:broadband:1-2:nemo:services:configuration"
id="2">
  <nemoc:NodeInfo>
    <!-- Registration Service Node's Encryption Key -->
    <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
```

```
protocol/basic/1.0#request-encryptionKey">
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
     <!-- Registration Service Node's Signing Key -->
     <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-signingKey">
        <wsse:Embedded>…..</wsse:Embedded>
      </wsse:SecurityTokenReference>
      <!-- Registration Service's Role Assertion -->
      <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2004/attribute/role">
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
  </nemoc:NodeInfo>
  <!-- Registration Service's WSDL -->
  <wsdl:definitions>…</wsdl:definitions>
  <!-- Registration Service's Policy-->
  <PolicyURI
operationName="nodeAcquisition">urn:marlin:broadband:1.0:nemo:services:registration-
service:nodeAcquisition:policy:0</PolicyURI>
  <PolicyURI
operationName="linkAcquisition">urn:marlin:broadband:1.0:nemo:services:registration-
service:linkAcquisition:policy:0</PolicyURI>
  <PolicyURI
operationName="deregistration">urn:marlin:broadband:1.0:nemo:services:registration-
service:deregistration:policy:0</PolicyURI>
</RegistrationServiceConfig>
```

5.3.1.3 Data Certification Service Configuration

A <bsc:DataCertificationServiceConfig> element corresponds to a Data Certification Service. This element MUST include the following information:

- <nemoc:NodeInfo>: signing, encryption NEMO keys, and role assertion information for the Data Certification Standard Service.
- <wsdl:definitions>: WSDL definitions for the DCS.
- <bsc:PolicyURI>: the Policy URI which is applied to the operation of data certification service wsdl. The operation is identified by.operationName attribute. As the Policy URI, the Data Certification Service policy defined in §5.2.3.4 SHALL be specified.
- <bsc:CertificationStandard>: For each data certification standard for which the DCS is authorized to issue DCSAs, the name of the certification standard and the set(s) of namespaces and names of the security-related metadata that are represented by the data certification standard.

Here is an example of a <DataCertificationServiceConfig> element:

```
<DataCertificationServiceConfig xmlns="urn:marlin:broadband:1-
2:nemo:services:configuration">
  <nemoc:NodeInfo>
    <!-- DCS Node's Encryption Key -->
    <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-encryptionKey">
```

```
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
     <!-- DCS Node's Signing Key -->
     <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-signingKey">
        <wsse:Embedded>…..</wsse:Embedded>
      </wsse:SecurityTokenReference>
      <!-- DCS's Role Assertion -->
      <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2004/attribute/role">
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
  </nemoc:NodeInfo>
  <!-- DCS's WSDL -->
  <wsdl:definitions>…</wsdl:definitions>
  <!-- DCS's Policy-->
  <PolicyURI operationName="
requestDataCertification">urn:marlin:broadband:1.0:nemo:services:datacertification-
service:policy:0</PolicyURI>
  <!-- DCSA provided by the DCS -->
  <CertificationStandard name="urn:marlin:organization:foo:bar">
    <dcsi:DataItem name="trusted-time" namespace="urn:marlin:broadband:security-
metadata:attributes"/>
    <dcsi:DataItem name="urn:marlin:organization:acme"
namespace="urn:marlin:broadband:security-metadata:attributes:license-suspension-
serial"/>
  </CertificationStandard>
</DataCertificationServiceConfig>
```

5.3.1.4 Data Update Service Configuration

A <bsc:DataUpdateServiceConfig> element corresponds to a Data Update Service. This element MUST include the following information:

- <nemoc:NodeInfo> signing, encryption NEMO keys, and role assertion information for the DUS.
- <wsdl:definitions>: WSDL definitions for the DUS.
- <bsc:PolicyURI>: the Policy URI which is applied to the operation of data update service wsdl. The operation is identified by.operationName attribute. As the Policy URI, the Data Update Service policy defined in §5.2.4.4 SHALL be specified.
- The namespaces and names of security-related metadata items that can be provided by the DUS.

Here is an example of a <DataUpdateServiceConfig> element:

```
<DataUpdateServiceConfig xmlns="urn:marlin:broadband:1-
2:nemo:services:configuration">
  <nemoc:NodeInfo>
    <!-- DUS Node's Encryption Key -->
    <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-encryptionKey">
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
    <!-- DUS Node's Signing Key -->
    <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-signingKey">
       <wsse:Embedded>…..</wsse:Embedded>
     </wsse:SecurityTokenReference>
     <!-- DUS’s Role Assertion -->
     <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2004/attribute/role">
     <wsse:Embedded>…..</wsse:Embedded>
   </wsse:SecurityTokenReference>
  </nemoc:NodeInfo>
  <!-- DUS’s WSDL -->
  <wsdl:definitions>…</wsdl:definitions>
  <!-- DUS’s Policy-->
  <PolicyURI operationName="
dataUpdataRequest">urn:marlin:broadband:1.0:nemo:services:dataupdate-
service:policy:0</PolicyURI>
  <!—Security-related metadata provided by the DUS -->
  <dcsi:DataItem name="trusted-time" namespace="urn:marlin:broadband:security-
metadata:attributes"/>
  <dcsi:DataItem name="urn:marlin:organization:acme"
namespace="urn:marlin:broadband:security-metadata:attributes:license-suspension-
serial"/>
</DataUpdateServiceConfig>
```

5.3.1.5 Metering Data Service Configuration

A <bsc:MeteringDataServiceConfig> element corresponds to a Metering Data Service. This element MUST include the following information:

- <nemoc:NodeInfo>: signing, encryption NEMO keys, and role assertion information for the MDS.
- <wsdl:definitions>: WSDL definitions for the MDS.
- <bsc:PolicyURI>: the Policy URI which is applied to the operation of metering data service wsdl. The operation is identified by.operationName attribute. As the Policy URI, the Metering Data Service policy defined in §5.2.5.4 SHALL be specified.
- <bsc:Namespace>: the namespace(s) for service-specific URI(s), which are recorded with the metering information. The Metering Data Service retrieves the metering information for the namespace(s).

Here is an example of a <MeteringDataServiceConfig> element:

```
<MeteringDataServiceConfig xmlns="urn:marlin:broadband:1-
2:nemo:services:configuration">
  <nemoc:NodeInfo>
   <!-- MDS’s Encryption Key -->
   <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-encryptionKey">
     <wsse:Embedded>…..</wsse:Embedded>
   </wsse:SecurityTokenReference>
    <!-- MDS’s Signing Key -->
    <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
```

```
protocol/basic/1.0#response-signingKey">
        <wsse:Embedded>…..</wsse:Embedded>
      </wsse:SecurityTokenReference>
      <!-- MDS's Role Assertion -->
      <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2004/attribute/role">
      <wsse:Embedded>…..</wsse:Embedded>
    </wsse:SecurityTokenReference>
  </nemoc:NodeInfo>
  <!-- MDS's WSDL -->
  <wsdl:definitions>…</wsdl:definitions>
  <!-- MDS's Policy-->
  <PolicyURI operationName="
meteringCollection">urn:marlin:broadband:1.0:nemo:services:metering-
service:policy:0</PolicyURI>
  <!-- Metering namespace supported by the MDS -->
  <Namespace>urn:marlin:organization:foo</Namespace>
  <Namespace>urn:marlin:organization:bar</Namespace>
</MeteringDataServiceConfig>
```

5.3.2 Action Token

An Action Token, which is expressed in a <bsa:ActionToken> element, MUST include resource location information for the Configuration Token and MUST also include relatively dynamic information that is necessary to communicate with Marlin BB services. The ResourceLocation element value SHALL be accessed using the HTTP GET method. An Action Token includes the following:

- <bsa:ConfigurationInfo>: The ConfigurationInfo element has two mandatory attribute:
    - broadbandServiceId: points to the broadbandServiceId attribute of the Marlin Configuration Token pointed by this ConfigurationInfo element
    - configVersion: points to the version of this Configuration.
- A sequence of one or more actions (e.g., <bsa:LicenseAcquisition>, <bsa:LinkAcquisition>, etc.)

The id attribute for Service Configuration is REQUIRED only if there are multiple License Service Configurations or Registration Service Configurations to be referred from Action Token. For other service configurations for which there is no corresponding Action Token, id attribute is not necessary.

The sequence of actions to be performed by the client receiving the Action Token is specified by the appearance order of the actions in the <ActionToken> element.

When a Marlin BB Service requires one or more DCSAs for its service request, the corresponding certification standard name(s) SHOULD be specified in each of the actions in the <bsa:CertificationStandard> child element. A <CertificationStandard> element includes the following attributes:

- name: This REQUIRED attribute specifies a certification standard name .
- use: This REQUIRED attribute specifies the use of the DCSA. The value of the attribute is either “must” or “should”. A value of “must” means that the DCSA is required by the service for its service request. A value of “should” means that it is recommended that the DCSA be sent for its service request. In this case,

even if the client does not send the DCSA in its service request, the service might accept the request, depending on the service's policy.

- validity: This OPTIONAL attribute defines the maximum acceptable age of the DCSA the service requires for its service request.

A copy of the XML schema for an Action Token is in Appendix A.7.

5.3.2.1 License Acquisition

Inclusion of a <bsa:LicenseAcquisition> element in a <bsa:ActionToken> element indicates that the client receiving the <ActionToken> should contact the License Service to obtain a License. A <LicenseAcquisition> element MUST include the following information:

- <Uid> or <Type>: Either the Uid or the Type of the Octopus Node to which the License will be bound. The Type is only used to indicate personality (i.e., it indicates that the Node is an Octopus Personality Node).
- A <bsa:BusinessToken> containing service-specific data.

Here is an example of a <LicenseAcquisition> element:

```
<LicenseAcquisition xmlns="urn:marlin:broadband:1-2:nemo:services:action-token" id="1">
   <Type>personality</Type>
   <BusinessToken>UjBsR09EbGhjZ0d</BusinessToken>
   <CertificationStandard
name="urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME" use="must"
validity="P1M"/>
</LicenseAcquisition>
```

Note: The "CERTIFICATION_STANDARD_NAME" is a uniquely-defined value within the organization URI.

5.3.2.2 Node Acquisition

Inclusion of a <bsa:NodeAcquisition> element in a <bsa:ActionToken> element indicates that the client receiving the <ActionToken> should contact the Registration Service to obtain a Node. A <NodeAcquisition> element MUST include the following information:

- A <bsa:BusinessToken> containing service-specific data.

Here is an example of a <NodeAcquisition> element:

```
<NodeAcquisition xmlns="urn:marlin:broadband:1-2:nemo:services:action-token" id="2">
   <BusinessToken>UjBsR09EbGhjZ0d</BusinessToken>
   <CertificationStandard
name="urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME" use=”should"
validity="P1M"/>
</NodeAcquisition>
```

5.3.2.3 Link Acquisition

Inclusion of a <bsa:LinkAcquisition> element in a < bsa:ActionToken> element indicates that the client receiving the <ActionToken> should contact the Registration Service to obtain a Link Object. A <LinkAcquisition> element MUST include the following information:

- <Uid> or <Type>: Either the Uid or the Type of the Octopus Node that will correspond to the <oct:LinkFrom> element in the Link Object that will be acquired. The Type is only used to indicate personality (i.e., it indicates the Node is an Octopus Personality Node).
- <Uid>: The Uid of the Octopus Node that corresponds to the <oct:LinkTo> element in the Link Object.
- A <bsa:BusinessToken> containing service-specific data.

Here is an example of a <LinkAcquisition> element:

```
<LinkAcquisition xmlns="urn:marlin:broadband:1-2:nemo:services:action-token" id="2">
   <Type>personality</Type>
   <Uid>urn:sample:user:00000001</Uid>
   <BusinessToken>UjBsR09EbGhjZ0d</BusinessToken>
   <CertificationStandard
name="urn:marlin:organization:acme:CERTIFICATION_STANDARD_NAME" use="should"
validity="P1D"/>
</LinkAcquisition>
```

5.3.2.4 Deregistration

Inclusion of a <bsa:Deregistration> element in a <bsa:ActionToken> element indicates that the client receiving the <ActionToken> should contact the Registration Service to deregister. A <Deregistration> element MUST include the following information:

- <Uid> or <Type>: Either the Uid or the Type of the Octopus Node that corresponds to the <oct:LinkFrom> element in the Link Object whose link will be broken by the deregistration. The Type is only used to indicate personality (i.e., it indicates the Node is an Octopus Personality Node).
- <Uid>: The Uid of the Octopus Node that corresponds to the <oct:LinkTo> in the Link Object.
- A <bsa:BusinessToken> containing service-specific data.

Here is an example of a <Deregistration> element:

```
<Deregistration xmlns="urn:marlin:broadband:1-2:nemo:services:action-token" id="2">
   <Type>personality</Type>
   <Uid>urn:sample:user:00000001</Uid>
   <BusinessToken>UjBsR09EbGhjZ0d</BusinessToken>
</Deregistration>
```

5.3.3 Processing Rules for Configuration and Action Tokens

When a client is required to get one or more DCSAs (each of which is represented by a particular data certification standard name) to access a particular Marlin BB service or License, corresponding DCS Configuration(s) SHALL be provided by the Configuration Token. The DCS Configuration(s) SHALL contain the data certification standard name(s) and the names and namespaces of the security-related metadata represented by the data certification standard name. This enables a client to know which DCSA can be acquired from which DCS. Furthermore, this also enables a client to know which security-related metadata is required in order to acquire a given DCSA.

When a client is required to get one or more DCSAs, which in turn will require the client to update security-related metadata from one or more DUSs if the client metadata is not adequately up-to-date, corresponding DUS Configuration(s) SHALL be provided by the Configuration Token. The (set of) DUS Configuration(s) SHALL be able to provide all the security-related metadata items necessary for a client to acquire the DCSA(s). This enables a client to know which DUS can be used to acquire a given security-related metadata item.

When a Marlin BB service specifies, via a <bsa:CertificationStandard> element in an action in the <bsa:ActionToken>, that a particular DCSA is required, the client can determine which DCS will provide the corresponding DCSA from information in the <bsc:DataCertificationService> element(s) in the Configuration Token. Additionally, each <DataCertificationService> specifies which security-related metadata is required in order to obtain the DCSA. A client MAY retrieve necessary security-related metadata from the DUS, before accessing the DCS.

When a validity attribute is specified in a <CertificationStandard> element in an action in the <ActionToken>, a client MAY determine whether its previously acquired corresponding DCSA is valid for the Marlin BB service. When it determines that its DCSA is no longer valid for the Marlin BB service, a client MAY try to acquire a corresponding new DCSA before accessing the intended Marlin BB service.

5.3.4 MIME Type Definitions

The following table defines the MIME types for Configuration Tokens, Action Tokens and License bundles:

File	**MIME type**
Configuration Token	application/vnd.marlin.drm.conftoken+xml
Action Token Set	application/vnd.marlin.drm.actiontoken+xml
License Bundle	application/vnd.marlin.drm.license+xml

6 Broadband-specific usage rules

6.1 Introduction

Broadband usage models introduce additional usage rules not currently defined in [MCS]:

- License Suspension, to support such actions as invalidating erroneously distributed contents (or corresponding licenses), or invalidating contents of artists whose contracts are no longer in force
- Metering, for accumulating and reporting content usage

These additional features can be referenced in licenses destined to Marlin BB-conformant devices and MAY be required to consume the subscription contents.

6.2 License Suspension

6.2.1 License Identification

Any Licenses that the issuer wishes to be able to be suspended MUST each be given a *logical Id* by the License Service. The “logical Id” MAY be coded in a Plankton (see [8pus] §4) bytecode or as an attribute of the Control object. This id is represented as a string, and SHOULD NOT be interpreted in any way by the client.

Multiple Licenses MAY be given the same “logical Id”, with the effect that, once suspended, all those Licenses MUST NOT be usable until the suspension is released. Alternatively, Licenses that are logically equivalent, but issued to different users, MAY be given a unique logical Id. Thus, the logical Id can represent a variety of concepts and categorizations.

The License Service MAY add checks in the Control of the License to ensure that the “logical Id” does not appear on the master License Suspension List that is maintained in the client. If the logical Id in the Control does appear on the master License Suspension List, then the Control MUST evaluate to false and not grant access to the content.

6.2.2 License Suspension Lists

A client which supports license suspension MUST securely maintain a master License Suspension List that is logically composed of multiple License Suspension Lists associated with a service-specific URN (namespace).

The set of Licenses that are considered suspended will change over time. The client MUST support incremental updates to this list by obtaining License Suspension Updates from the DUS. An update of the suspension lists MAY be required in order for the client to obtain a valid DCSA.

6.2.3 License Checks

Enforcement of suspensions is accomplished by encoding in the Control a check for license suspension. The System.Host.GetObject Plankton system call is used for checking for License Suspension.

The following is the object path for License Suspension checks.

Object Path
/Marlin/LicenseSuspension/<service container name>/IdList/<logical Id>

The implementation of System.Host.GetObject for License Suspension SHALL interpret a request for a host object under /Marlin/LicenseSuspension as being a request to check for the presence of the specified logical Id in the master License Suspension List under the <service container name> which corresponds to the value of the namespace attribute in the License Suspension Update.

As an example, the following object path would be used to check whether the logical Id “Jazz-23456” appears on the master License Suspension List under the namespace “urn:marlin:organization:acme”.

/Marlin/LicenseSuspension/urn:marlin:organization:acme/IdList/Jazz-23456

System.Host.GetObject SHALL return an integer host object whose value is 1 (signifying true) if the logical Id has been suspended. Otherwise, it SHOULD return the error code ERROR_NO_SUCH_ITEM.

6.2.4 License Suspension Updates

The master License Suspension List (maintained securely by the client in some implementation-specific way) is updated by License Suspension Update.

The <dus:LicenseSuspensionUpdate> element includes the namespace (in the namespace attribute), the Serial Number (serial attribute), a Reset Flag (resetBeforeConsumption attribute), a <dus:Subtractions> set of logical Ids (<dus:LogicalID>) to subtract from the master License Suspension List, and a <dus:Additions> set of logical Ids to add to the master License Suspension List. Either the <Subtractions> or the <Additions> list can be empty.

Copies of the XML schema and the WSDL are in Appendices A.4 and B.4, respectively.

The master License Suspension List is partitioned into namespaces, which are service-specific URNs. A given License Suspension Update, which specifies the namespace associated with the update, MUST only impact the specific master License Suspension List partition identified by the namespace.

Each License Suspension Update MUST be marked with a Serial Number (serial attribute), and the client MUST maintain (securely and persistently) the highest serial number value it has processed for a given namespace.

The resetBeforeConsumption attribute (i.e., the Reset Flag) is an OPTIONAL attribute used to reset the master License Suspension List for a given namespace. When the resetBeforeConsumption attribute is set to true, the master License Suspension List for the given namespace MUST be deleted before the License Suspension Update is processed. When the attribute is not present or is set to false, the master License Suspension List MUST NOT be reset.

For a given namespace, when multiple License Suspension Updates are included in a response message of DUS, License Suspension Updates for the namespace SHALL be sorted by serial number from the lowest to the highest serial number in the response message.

If a License Suspension Update includes inconsistent settings (e.g., subtractions of nonexistent logical id(s) from the master License Suspension List), the client SHALL ignore inconsistent settings and continue processing the remaining settings in the License Suspension Update.

Here is a License Suspension Update example:

```
<LicenseSuspensionUpdate xmlns="urn:marlin:broadband:1-2:nemo:services:schemas:data-
update-service" namespace="urn:marlin:organization:acme" serial="2"
resetBeforeConsumption=”false”>
   <Subtractions>
      <LogicalId>883</LogicalId>
   </Subtractions>
   <Additions>
      <LogicalId>2020202</LogicalId>
      <LogicalId>foobar</LogicalId>
      <LogicalId>Jazz-23432</LogicalId>
   </Additions>
</LicenseSuspensionUpdate>
```

The intent of the License Suspension Update in the above example is to cause the subtraction of the logical Id 883 from the current master License Suspension List (under the namespace “urn:marlin:organization:acme”) and the addition of the logical Ids 2020202, foobar, and Jazz-23432 to the list.

6.3 Obligation for Metering

The obligation mechanism defined in [8pus] §3.4 is used to signal a requirement for metering. An obligation for metering allows a license to express the requirement that the application capture and report usage data compliant with this specification. The primary motivation for the metering data is to support the financial viability of subscription services. Content providers provide better financial terms for use of content under a subscription model if metering data is available.

The following obligation is defined for metering.

Obligation	Argument
urn:marlin:broadband:obligation:meter-play-duration	namespace
	logical-id

The namespace is a service-specific URN that identifies the service to which the metering data SHOULD be delivered. The logical-id is a service-defined identifier that MUST be recorded with the metering information for the content being played to allow the service to correlate the information as it sees fit.

The recorded play durations MUST include all the time during which content is rendered and presented to the user in a normal manner (normal play). The recorded duration SHOULD NOT include any other time. While desirable (to minimize royalty payments), it is not required that time for transient operations that are not “normal play” (e.g., fast forward or rewind) be excluded from the recorded duration. Long operations that are not “normal play” (e.g., pausing for an hour) SHOULD NOT be included in the recorded duration.

This obligation imposes several requirements on the application:

- Record the play duration in seconds using the provided namespace and logical-id. (When a client does not have enough resources, the client may record start and stop time of play instead of the play duration. In this case, an entity which receives the metering data is responsible for analyzing the data to determine the distinct play durations.)
- Provide integrity protection of the metering data until it is delivered to the service.
- Provide replay protection of the metering data.
- Provide reasonable measures to ensure eventual delivery of metering data to the service.
- Deliver the metering data in a timely manner.
- Protect the integrity of the metering data during delivery.
- Ensure user anonymity.

The recorded metering data is delivered to the Metering Data Service (MDS), partitioned by each of the namespaces provided as arguments in the obligation. That is, there is an <mds:MeteringData> element provided for each namespace. There are two possible record formats that can be specified in each <MeteringData> element, one that is used when play duration is recorded, and one that is used when start and stop times are recorded. These formats are described in the following sections.

6.3.1 Duration Record

A <mds:DurationRecord> element is used when a client records the play duration. This element MUST include the following information:

- A <mds:LocalTimestamp> or <mds:SecureTimestamp> specifying when the recording was begun. A <SecureTimestamp> MUST always be used if the device has a secure clock.
- The <mds:LogicalId> corresponding to an argument in the obligation.
- The <mds:Duration> of the play, in seconds.

Here is an example of a <MeteringData> element that reports play durations:

```
<MeteringData xmlns="urn:marlin:broadband:1-1:nemo:services:schemas:metering-
service" namespace="urn:marlin:organization:acme">
  <DurationRecord>
```

```
    <LocalTimestamp>2005-12-17T09:30:47Z</LocalTimestamp>
    <LogicalId>foobar</LogicalId>
    <Duration>PT1H30M0S</Duration>
  </DurationRecord>
  <DurationRecord>
    <LocalTimestamp>2001512-18T09:30:40Z</LocalTimestamp>
    <LogicalId>Jazz-23432</LogicalId>
    <Duration>PT30M0S</duration>
  </DurationRecord>
</MeteringData>
```

6.3.2 Event Record

An <mds:EventRecord> element is used when a client records start and stop time of play. This element MUST include the following information:

- A <mds:LocalTimestamp> or <mds:SecureTimestamp> specifying when the recording was begun or stopped (depending on whether a start or stop is specified as content of <mds:Event>). A <SecureTimestamp> MUST always be used if the device has a secure clock.
- The <mds:LogicalId> corresponding to an argument in the obligation.
- An <Event> element with value “start” or “stop” specifying whether the record was written in response to start of play or end of play, respectively.

Here is an example of a <MeteringData> element that reports start and stop events:

```
<MeteringData  xmlns="urn:marlin:broadband:1-1:nemo:services:schemas:metering-
service" namespace="urn:marlin:organization:acme">
  <EventRecord>
    <LocalTimestamp>2005-12-17T09:30:47Z</LocalTimestamp>
    <LogicalId>foobar</LogicalId>
     <Event>start</Event>
   </EventRecord>
   <EventRecord>
     <LocalTimestamp>2005-12-17T10:30:47Z</LocalTimestamp>
     <LogicalId>foobar</LogicalId>
      <Event>stop</Event>
   </EventRecord>
  <EventRecord>
    <local-timestamp>2005-12-18T09:30:47Z</local-timestamp>
    <logical-id>Jazz-23432</logical-id>
     <Event>start</Event>
   </EventRecord>
   <EventRecord>
     <LocalTimestamp>2005-12-18T10:30:47Z</LocalTimestamp>
     <LogicalId>Jazz-23432</LogicalId>
      <Event>stop</Event>
   </EventRecord>
</MeteringData></MeteringData>
```

6.3.3 No Record

A <mds:NoRecord> element is used when a client cannot send the metering data to the MDS. It includes a reason attribute specifying one of the following values indicating the reason:

- norecord indicates there is no recorded data in the client for the specified namespace.
- recordfalsified indicates that the client detected that the recorded data was modified.

Here is an example of a <MeteringData> element that includes a <NoRecord> element:

```
<MeteringData xmlns="urn:marlin:broadband:1-1:nemo:services:schemas:metering-
service" namespace="urn:marlin:organization:acme">
  <NoRecord reason="norecord"/>
</MeteringData>
```

7 DRM Usage profiles for Marlin BB

7.1 Role Assertion

There are many roles in Marlin, and each role is represented by a distinct role assertion. In the context of a given role, there are OPTIONAL attributes in Marlin BB.

Just as a role assertion reflects the capabilities of the associated implementation component, in Marlin BB, a role assertion also reflects the identity and version of the component and is used to force renovation of components that are deemed compromised.

In Marlin BB, a number of additional attributes are defined for the urn:marlin:core:role:drm-client role.

7.1.1 Supported Marlin BB Specification Version

In Marlin BB, two additional Security Specification Version attributes are defined for the urn:marlin:core:role:drm-client role to signal. Refer to [MCS] §8.2 and §12.5.4.3 for a description of the semantics and renewability requirements of these trusted attributes.

- The version-major corresponds to the major version of Marlin BB Security Specification Version the client implements.
- The version-minor corresponds to the minor version of Marlin BB Security Specification Version the client implements.

Namespace	Name	Object Path
urn:marlin:broadband	version-major	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:version-major
urn:marlin:broadband	version-minor	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:version-minor

7.1.2 Client Capabilities

In Marlin BB, two additional attributes are defined for the urn:marlin:core:role:drm-client role to signal the client capabilities.

Namespace	Name	Object Path
urn:marlin:broadband:client:capabilities	license-suspension	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:client:capabilities:license-suspension
urn:marlin:broadband:client:capabilities	meter-play-duration	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:client:capabilities:meter-play-duration

The license-suspension is OPTIONAL capability for Marlin BB DRM Client. The presence of the license-suspension attribute indicates the device supports License Suspension.

The meter-play-duration is OPTIONAL capability for Marlin BB DRM Client. The presence of the meter-play-duration attribute indicates the device supports metering.

7.1.3 Manufacturer, Model, and Version

In Marlin BB, the following three additional attributes are defined for the urn:marlin:core:role:drm-client role.

Namespace	Name	Object Path
urn:marlin:broadband:client	manufacturer	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:client:manufacturer
urn:marlin:broadband:client	model	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:client:model
urn:marlin:broadband:client	version	/Marlin/Assertions/@<index>/Attributes/urn:marlin:broadband:client:version

The manufacturer attribute is a manufacturer-specific URN which indicates the namespace of the values specified for the model and version attributes. In other words, the model and version values MUST be defined under the management of the URN specified by the manufacturer attribute. The version itself MAY be specified either as a string or as a container with version component names (e.g., major, minor, revision).

The manufacturer, model, and version attributes SHALL NOT be used to preclude access to content by legitimate device manufacturer’s models.

Control programs SHOULD NOT reference the manufacturer, model, and version attributes.

Renewability can be encouraged by shunning compromised clients. The DCS MUST refuse to issue the necessary assertions to any such compromised client. Other services (e.g., license acquisition, registration, etc.) MUST also refuse to provide service when the client’s role assertion identifies the underlying implementation as one that has been deemed compromised.

Appendix A XML Schemas File Names

A.1 License.xsd

A.2 Registration.xsd

A.3 DataCertification.xsd

A.4 DataUpdate.xsd

A.5 Metering.xsd

A.6 Broadband-services-config.xsd

A.7 Broadband-services-action.xsd

A.8 DataCertificationDataItem.xsd

Appendix B WSDLs File Names

B.1 License.wsdl

B.2 Registration.wsdl

B.3 DataCertification.wsdl

B.4 DataUpdate.wsdl

B.5 Metering.wsdl