ISBM 2.1-RC1

Implementation Specification for ISA-95 Message Service Model

OpenO&M Specification

2023-04-22

Editors

MIMOSA

Matt Selway, University of South Australia
Karamjit Kaur, University of South Australia

ISA

Dennis Brandl, BR&L Consulting
Douglas Brandl, BR&L Consulting

Status

Implementors should be aware that this specification is not stable. *Implementors who are not taking part in the* discussions *are likely to find the specification changing out from under them in incompatible ways.* Vendors interested in implementing this specification before it eventually reaches the final Standard stage should join the MIMOSA mailing lists (or equivalent) and take part in the discussions through the OpenO&M Working Group.

The latest stable version of the editor's draft of this specification is always available on the MIMOSA ISBM Git repository [https://github.com/mimosa-org/isbm].

If you wish to make comments regarding this specification in a manner that is tracked by OpenO&M, please submit them via the public bug database [https://github.com/mimosa-org/isbm/issues]. You can alternatively contact MIMOSA directly [http://www.mimosa.org/contact] and arrangements will be made to transpose appropriate remarks to the public bug database. All feedback is welcome.

Latest Version

This is version 2.1 which can be found at: http://www.openoandm.org/isbm/2.1
The latest published version of this specification can always be found at: http://www.openoandm.org/isbm/latest

Notices

All capitalized terms in the following text have the meanings assigned to them in the MIMOSA Intellectual Property Rights Policy (the "MIMOSA IPR Policy"). The full Policy may be found at the MIMOSA website [http://www.mimosa.org/policy-charters/mimosa-intellectual-property-rights-policy/].

The OpenO&M ISBM is released under the MIMOSA License Agreement [http://www.mimosa.org/policy-charters/mimosa-license-agreement/].

Contents

Foreword

This document defines a SOAP Web Service implementation of the ISA 95.00.06 Messaging Service Model (MSM) as well as describing a plain HTTP/JSON REST interface defined by the OpenO&M ISBM Joint Working Group (JWG).

This revision aims to ensure support for secure inter-enterprise communication via minimal extensions in-line with revisions to ISA 95.00.06 MSM.

OpenO&M is an initiative of multiple industry standards organizations to provide a harmonized set of standards for the exchange of Operations & Maintenance (O&M) data and associated context. OpenO&M is an open, collaborative, effort composed of diverse groups of relevant organizations and subject matter experts. The members of OpenO&M initiative include ISA, MESA, MIMOSA, OAGi, and the OPC Foundation.

- MIMOSA provides asset management related information standards
- ISA provides industrial automation standards
- OPC Foundation provides data acquisition and transport standards

Participating organizations work together to cross-reference their related standards, collaborate on the content and where possible to incorporate each-others work by reference, with the objective of providing a foundation for standards-based interoperability.

The specification described in this document is an implementation specification as opposed to a standard. This specification is validated for ease of implementation and use via reference implementations made available by the OpenO&M ISBM JWG members, e.g., the OpenMSM [https://github.com/OpenMSM/OpenMSM] and ProtoISBM [https://github.com/mattys101/ProtoISBM], but this does not preclude commercial implementations from being developed to conform to this specification.

The features and capabilities of the specification are validated through the OIIE OGI Pilot [http://www.mimosa.org/ogi-pilot/] to ensure that it is fit-for-purpose. The OIIE OGI Pilot also provides feedback into this specification for new features, capabilities, and requirements that are considered in future revisions of the specification.

Introduction

This specification defines a SOAP Web Service and a HTTP/JSON REST implementation of the ISA-95.00.06 Messaging Service Model (MSM).

Its purpose is to provide additional specificity that is required to enable two or more groups to develop implementations of the MSM that will properly interoperate with each other without a priori knowledge of each other. It provides a consistent set of specifications supporting both intra and inter-enterprise activities, where a combination of functionality, security, supplier-neutrality and ease of implementation are required for industry digital transformation.

This specification defines a minimal interface subset to message exchange middleware using standard Web Service and REST interfaces. Publish-subscribe and request-response messaging patterns are supported through a consistent and unified model. Message routing is conducted through shared channels and topics, and optionally, XPath/JSONPath filtering for granular content-based filtering. An asynchronous Web Service callback or an asynchronous callback REST service is also provided to clients for notification of received messages. Token-based security for channels is specified to support multiple authorization models, from basic credential exchange to federated identity providers.

In addition to the services defined by ISA-95.00.06 MSM, this specification includes services for configuration discovery to allow applications to determine their compatibility with a service provider.

This specification also considers the security implications of the ISBM that may arise in different organizational contexts, such as intra- or inter-enterprise contexts. It describes several levels of security, the sets of features required to conform to each level, and the contexts in which they are considered most appropriate.

The benefit of this implementation specification is that it allows applications to expose a single, standardized interface instead of a custom-built interface for every version and format of message exchange systems. It also allows applications to select REST or Web Services based on the application requirements. The goal is to further interoperability in application to application communications.

ISBM 2.1

1 Scope

This is an implementation specification of a set of Web Services for the messaging services described in ISA-95.00.06 Messaging Service Model (MSM) and related specifications for configuration discovery and security. The Web Services are defined for both SOAP (1.1 and 1.2) and as a RESTful API.

2 Normative References

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ANSI/ISA-95.00.06-2014, Enterprise-Control System Integration – Part 6: Messaging Service Model

IETF RFC 2616, Hypertext Transfer Protocol HTTP/1.1, 1999

W3C Recommendation, XML Schema Part 1: Structures Second Edition, 2004

W3C Recommendation, XML Schema Part 2: Datatypes Second Edition, 2004

ISO/IEC 21778:2017, The JSON data interchange syntax

OASIS Standard, Web Services Security: SOAP Message Security 1.0, 2004

3 Terms, Definitions, and Conventions

3.1 Terms

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at http://www.iso.org/obp
- IEC Electropedia: available at http://www.electropedia.org/

3.1.1
ChannelDescription
text that describes a channel

[SOURCE: ISA-95.00.06-2014, 3.1.1]

3.1.2
ChannelFault
error indicating that either the ChannelURI is invalid or the application does not have the appropriate SecurityToken to access the channel

3.1.3
ChannelType
primary use of a channel for publications or requests

[SOURCE: ISA-95.00.06-2014, 3.1.2]

3.1.4
ChannelURI
primary identifier for a channel

[SOURCE: ISA-95.00.06-2014, 3.1.3]

3.1.5
Expiry
duration until the expiration of a publication message on a publication channel

[SOURCE: ISA-95.00.06-2014, 3.1.7]

3.1.6
FilterExpression
filtering element that may be applied to messages on a channel

[SOURCE: ISA-95.00.06-2014, 3.1.4]

3.1.7
interface
definition of a set of operations that can be performed by a software system

EXAMPLE 1 The WSDL definition of the Channel Management Service would be its SOAP interface.

EXAMPLE 2 The OpenAPI definition of the Channel Management Service would be its REST interface.

3.1.8
ListenerURL
implementation defined element that is used to indicate to an application when a new message has arrived

[SOURCE: ISA-95.00.06-2014, 3.1.5]

NOTE 1 Used to indicate when a new message is available for a session.

3.1.9
MessageContent
body of the message

[SOURCE: ISA-95.00.06-2014, 3.1.6]

3.1.10
MessageID
identifier generated upon posting of a message to a channel in a session

[SOURCE: ISA-95.00.06-2014, 3.1.8]

3.1.11
Namespace
collection of names or words that define a formal and distinct set

[SOURCE: ISA-95.00.06-2014, 3.1.9]

3.1.12
NamespaceFault
error indicating that duplicate namespace prefixes occur in the namespace parameters

NOTE 1 Namespaces prefixes MUST be unique.

3.1.13
NamespaceName
name used for an XPath/JSONPath filter expression

3.1.14
NamespacePrefix
prefix used for an XPath/JSONPath filter expression

3.1.15
OperationFault
error indicating that the attempt to open a Session on a Channel is of the wrong ChannelType

NOTE 1 The channel type MUST be of Publication type or Request type

3.1.16
ParameterFault
error indicating that the parameter for an operation is malformed or not optional and blank

3.1.17
resource
entity identified by a URI for a REST interface

NOTE 1 In this context a resource is typically a Channel, Session, Message or a collection thereof.

3.1.18
SecurityToken
physical device or software code used to gain access to a channel

[SOURCE: ISA-95.00.06-2014, 3.1.10]

3.1.19
SecurityTokenFault
error indicating that an invalid SecurityToken is used

3.1.20
SessionFault
error indicating that Session being accessed is of the wrong SessionType

3.1.21
SessionID
identifier generated upon an application creating a session on channel and provided to the application for use in the MSM services

[SOURCE: ISA-95.00.06-2014, 3.1.11]

3.1.22
SessionType
the kind of application session determining the operations that may be performed by the application on the MSM services

NOTE 1 Defined SessionTypes are Publication Provider, Publication Consumer, Request Provider, and Request Consumer.

3.1.23
Topic
identification of the information content in a message

[SOURCE: ISA-95.00.06-2014, 3.1.12]

3.1.24
Web Service
software system designed to support interoperable machine-to-machine interaction over a network

[SOURCE: https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice]

NOTE 1 It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

NOTE 2 This definition only applies when this capitalization is used.

3.1.25
OriginalMessageID
Identifier provided when a received message is forwarded to another channel

3.1.26
ExpirationListenerURL
implementation defined element that is used to indicate to an application when a message has been expired

3.2 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [http://www.ietf.org/rfc/rfc2119.txt].

This specification uses the following syntax to define conceptual structures and schema elements:

```
Element Name (Type) [Cardinality]
```

The namespaces for Types are defined in the following section. For example, the Topic element defined as an XML Schema `string` with one to many cardinality would be defined as: `Topic (xs:string) [1..*]`.

3.3 Schema Namespaces

The following namespaces are used in this document:

Prefix	Namespace
xs	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
isbm	http://www.openoandm.org/isbm/
isbm-rest	http://www.openoandm.org/isbm/rest/
json	Used for the differentiation of basic data types

4 Service Requirements

The following items define shared requirements that are applicable across the various services defined in Service Definitions. These requirements supplement the service requirements specified by ISA-95.00.06 but are contextualized for SOAP Web Services and REST interfaces.

4.1 Message Content Format

Messages are exchanged using a format appropriate to the service interface, for example, XML for the SOAP Interface and JSON for the REST interface. To support transparency of the interface and environments with

mixed Web Service implementations, the Message Content MAY be a content type different to the type used to represent the message. For example, JSON Message Content in an XML message over the SOAP interface.

All ISBM service providers MUST support JSON and XML content types as Message Content.

The ISBM service providers MAY support other content types as Message Content.

If an ISBM service provider implements both the SOAP interface and REST interface, messages received via one interface MUST be able to be forwarded via the other interface.

NOTE While the Message Topic describes the specific format and schema of the Message Content, the message schemas themselves, see below, describe the MIME Type to ensure accurate processing of the Message Content.

4.1.1 SOAP Interface Requirements

The XML Schemas for the SOAP interface are defined such that they allow the exchange of XML, JSON, and other Message Content types within the XML SOAP messages. The XML schema for Message Content defined for the SOAP interface makes use of type inheritance to support the different content types: XML, String, and Binary.

For XML Message Content, the content is associated with a message through the use of an XML Schema `any` (`xs:any`) element with the processing requirement defined as lax (processContents="lax"). An ISBM Service Provider SHOULD preserve significant whitespace and comments within the XML content. An XML declaration MUST NOT appear within the XML Message Content.

For String Message Content, the content is associated through the use of an XML element of type `xs:string`. This allows content represented by textual data formats, such as JSON, to be exchanged within the XML message. The String content MUST have its `mediaType` specified. A list of media types is available from IANA [https://www.iana.org/assignments/media-types/media-types.xhtml]. The String content MUST be correctly escaped according to the XML syntax if it includes protected XML characters.

A SOAP-based ISBM Service Provider SHOULD NOT exchange XML Message Content using the String content type.

For Binary Message Content, the content is associated through the use of an XML element containing Base64 encoded data (`xs:base64binary`). This allows content represented by binary formats to be exchanged within the XML message. The Binary content MAY specify the `mediaType` of the content. A list of media types is available from IANA [https://www.iana.org/assignments/media-types/media-types.xhtml].

A SOAP-based ISBM Service Provider SHOULD NOT exchange XML Message Content using the Binary content type.

In an XML message, the XML Schema Instance '`type`' (`xsi:type`) attribute MUST be used to indicate the specific content type.

4.1.1.1 XML Message Content Example

```
<MessageContent xsi:type="XMLContent">
  <Property>There could be arbitrary XML content (with a single root node) included here.</Property>
</MessageContent>
```

4.1.1.2 String Message Content Example

```
<MessageContent xsi:type="StringContent" mediaType="application/json">
  <Content>
{
  "prop": "There could be a JSON message, or anything else really."
}
  </Content>
</MessageContent>
```

4.1.1.3 Binary Message Content Example

```
<MessageContent xsi:type="BinaryContent">
  <!-- strictly speaking there should be no newlines after/before the element tags below -->
  <Content>
ew0KwqDCoCJwcm9wIjrCoCJUaGVyZcKgY291bGTCoGJlwqBhwqBKU09OwqBtZXNz
YWdlLMKgb3LCoGFueXRoaW5nwqBlbHNlwqByZWFsbHkuIg0KfQ==
  </Content>
</MessageContent>
```

4.1.2 REST Interface Requirements

The JSON Schemas for the REST interface are defined such that they allow the exchange of XML, JSON, and other Message Content types within JSON messages. The JSON schema for Message Content defined for the REST interface makes use of a flexibly defined 'content' property to support the different content types: JSON, String, and Binary.

For JSON Message Content, the content is associated with a message through the use of a JSON object as the property value. The JSON content MUST be valid JSON. The JSON content MUST NOT specify a `mediaType` nor `contentEncoding`.

In specific implementations of this specification, the JSON content MAY specify the URL of a JSON Schema if the ISBM Service Provider is to validate the JSON content against a schema.

For String Message Content, the content is associated through the use of a `string` as the property value. This allows content represented by textual data formats, such as XML, to be exchanged within the JSON message. The String content MUST have its `mediaType` specified. A list of media types is available from IANA [https://www.iana.org/assignments/media-types/media-types.xhtml]. The String content MUST be correctly escaped according to the JSON syntax if it would include protected JSON characters.

A REST-based ISBM Service Provider SHOULD NOT exchange JSON Message Content using the String content type within a JSON message.

A REST-based ISBM Service Provider SHOULD NOT exchange XML Message Content using the String content type within an XML message.

For Binary Message Content, the content is associated through the use of a `string` as the property value and an additional `contentEncoding` property that specifies the encoding type, e.g., `base64`. This allows content represented by binary formats to be exchanged within the JSON message. The Binary content MUST specify the `contentEncoding` of the content. The `contentEncoding` value MUST be encoding types commonly supported by HTTP. The list of encoding types is available from IANA [https://www.iana.org/assignments/http-parameters/http-parameters.xhtml#content-coding], in addition to this list base64 can be used as the basic level of encoding for binary content. The Binary content MAY specify the `mediaType` of the (decoded) content. A list of media types is available from IANA [https://www.iana.org/assignments/media-types/media-types.xhtml]..

A REST-based ISBM Service Provider SHOULD NOT exchange JSON Message Content using the Binary content type within a JSON.

A REST-based ISBM Service Provider SHOULD NOT exchange XML Message Content using the Binary content type within an XML message.

The Channel URIs MUST be encoded when used within the URL of a REST call, for example: `'http://server/channels/%2Fencoded%2Fchannel%2FURI'`

4.1.2.1 JSON Message Content Example

The following is an HTTP request for the Post Publication operation containing JSON Message Content within a JSON message.

```
POST /sessions/321/publications HTTP/1.1
Host: http://example.com
Accept: application/json
Content-Type: application/json
Content-Length: 183

{
  "topics": ["topic1", "etc"],
  "expiry": "P1D",
  "messageContent": {
    "content": {
      "somejson": "This is some JSON native content"
    }
  }
}
```

4.1.2.2 String Message Content Example

The following is an HTTP request for the Post Publication operation containing XML content using the String Message Content type within a JSON message.

```
POST /sessions/321/publications HTTP/1.1
Host: http://example.com
Accept: application/json
Content-Type: application/json
Content-Length: 187

{
  "topics": ["topic1", "etc"],
  "expiry": "P1D",
  "messageContent": {
    "mediaType": "application/xml",
    "content": "<someXml>This is XML content in JSON</someXml>"
  }
}
```

4.1.2.3 Binary Message Content Example

The following is an HTTP request for the Post Publication operation containing XML content using the Binary Message Content type within a JSON message. The content would decode to the same as the String Message Content Example.

```
POST /sessions/321/publications HTTP/1.1
Host: http://example.com
Accept: application/json
Content-Type: application/json
Content-Length: 238

{
  "topics": ["topic1", "etc"],
  "expiry": "P1D",
  "messageContent": {
    "mediaType": "application/xml",
    "contentEncoding: "base64",
    "content": "PHNvbWVYbWw+VGhpcyBpcyBYTUwgY29udGVudCBpbiBKU09OPC9zb21lWG1sPg=="
  }
}
```

4.2 Security

Security in the ISBM specification only provides authorization of channels. Authorization of services is considered out-of-scope.

All ISBM implementations MUST support transport layer security (e.g. SSL/TLS) in order to secure tokens and messages, and to prevent replay attacks.

All ISBM implementations MUST support username/password authentication as a basic level of security. This will differ for implementations of the different service types: for example, WS-Security UsernameToken for the SOAP interface and HTTP basic or digest authentication for the REST interface.

A ISBM Service Provider MAY choose to support additional forms of security tokens (e.g., SAML assertions, OAuth tokens) and it is RECOMMENDED that a ISBM Service Provider support out-of-band token exchange standards such as SAML [http://saml.xml.org/saml-specifications], WS-Federation [http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html] or OAuth [http://oauth.net/].

An ISBM Service Provider MUST validate security tokens for every service operation except for the Channel Management Service CreateChannel operation (since the channel does not exist at the point in time when invoking CreateChannel). For the provider and consumer services, tokens are validated upon every operation to ensure that an application has valid credentials even after a session is opened (in the event of token revocation).

4.2.1 SOAP Interface Requirements

All ISBM SOAP implementations MUST support the WS-Security UsernameToken [https://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf] using `PasswordText` as a basic level of security token. Examples of its use can be found in Example HTTP Flows.

NOTE the requirement for supporting the use of WS-Security UsernameToken is in place of the, usually optional, SecurityToken listed as an input to most operations in the service descriptions of the ISA-95.00.06 MSM specification.

As security tokens in the Channel Management Service are specified using XML Schema `any` element, tokens MUST be able to be represented in an XML format. For tokens that do not have a canonical XML representation, an ISBM Service Provider MUST define the supported formats.

4.2.2 REST Interface Requirements

All ISBM REST implementations MUST support the standard HTTP/1.1 authentication [https://tools.ietf.org/html/rfc7235] and authorization headers with potential support for security tokens. The credentials will be compared to SecurityTokens associated with the channel. The REST security tokens can be the same tokens used in the SOAP interface.

NOTE the requirement for supporting HTTP/1.1 authentication and authorization headers is in place of the, usually optional, SecurityToken listed as an input to most operations in the service descriptions of the ISA-95.00.06 MSM specification.

All ISBM REST implementations MUST support the `basic` authentication scheme of HTTP. It is RECOMMENDED that the Bearer authentication scheme be supported to allow use of OAuth 2.0 [https://oauth.net/2/] and JWT (JSON Web Tokens) [https://tools.ietf.org/html/rfc7519], for example.

As security tokens in the Channel Management Service are specified using a JSON Object, tokens MUST be able to be represented in a JSON format. For tokens that do not have a canonical JSON representation, an ISBM Service Provider MUST define the supported formats. The `UsernameToken` schema defined in this specification MUST be supported by ISBM REST implementations.

4.3 Error Handling

Faults MUST have an accompanying human readable explanation. For a SOAP 1.1 implementation this is provided through the SOAP `faultstring` element (see SOAP 1.1, SOAP Fault [http://www.w3.org/TR/soap11/#_Toc478383507]). For a SOAP 1.2 implementation this is provided through the

SOAP `Reason` element (see SOAP 1.2, SOAP Reason Element [http://www.w3.org/TR/soap12-part1/#faultstringelement]).

For REST implementation, this is provided through a simple object schema containing the `fault` property (e.g., ParameterFault).

NOTE The declared Faults specified by the services do not have any elements or attributes defined (other than the `fault` property for the REST definitions). This is because the sender can interpret the fault based on the supplied parameters and/or the operation behavior. For example, a `ChannelFault` returned by the DeleteChannel operation means that the `ChannelURI` provided by the sender did not exist.

4.3.1 Parameter Faults

If any parameter for an operation is malformed or not optional and blank, then an ISBM Service Provider MUST return a `ParameterFault` to aid senders in determining the type of error.

For the SOAP interface, the undeclared `isbm:ParameterFault` element MUST be used in the fault details.

For the REST interface, the defined json:ParameterFault object MUST be used with a HTTP Response of 400 'Bad Request'.

The Fault MAY carry the offending parameter name/s and it is RECOMMENDED that the parameter names be included only in non-production environments, in order to eliminate information that may compromise security in production environment.

NOTE Parameter Faults are implicit for all Service Definitions and do not appear in the list of faults.

4.3.2 Invalid Notification URL

If a provider/consumer application provides a URL that does not host a NotifyListener service, the ISBM Service Provider MAY choose to defer or not to send (possibly after some time) a NotifyListener request considering intermittent network issues.

NOTE If a provider/consumer application provides a malformed URL, a `ParameterFault` is returned. An invalid Notification URL is one in which the address is unreachable.

4.4 Content-Based Filtering

To allow efficient content-based filtering of messages, `FilterExpression`/s MAY be added to a subscription or read request session to provide a filtering definition. As part of a conforming specification an ISBM Service Provider MUST declare the expression languages for which it provides support (possibly none). An ISBM Service Provider MUST ignore any `FilterExpression`/s not specified in a supported expression language by treating the expression to be defined using the special 'ALLOW-ALL' expression language.

A special expression language 'ALLOW-ALL' MUST be supported by all ISBM Service Providers that support content-based filtering. When the expression language is 'ALLOW-ALL' the expression MAY be empty. The result of evaluating an expression of the 'ALLOW-ALL' language MUST always return the complete MessageContent.

It is RECOMMENDED that an XPath expression be used for XML content and a JSONPath [https://goessner.net/articles/JsonPath/] expression be used for JSON content. Other valid expression languages MAY also be used.

An XPath expression MUST be defined as an XPath v1.0 expression.

The `FilterExpression` MAY specify the `mediaType`/s to which the expression applies. If no `mediaType` is specified, the expression MUST be applied to Message Content of all content types.

More than one `FilterExpression` MAY be added to a subscription or read request session to allow different expressions, expressions in different languages, or expressions for different `mediaType`/s to be specified. For example, an XPath expression for XML content and a JSONPath expression for JSON content.

Only one `FilterExpression` for any particular combination of expression language and `mediaType` SHOULD be allowed.

If a `FilterExpression` is present, then a notification MUST NOT be generated, and the message MUST NOT be made available to the receiving system under any of the following conditions:

- The `FilterExpression`'s `mediaType` matches that of the Message Content and evaluation of the expression returns an empty value or node set (or otherwise is considered to not match the content based on the rules of the expression language); or
- There is no expression with a `mediaType` that matches that of the Message Content; or
- The evaluation of the expression is not possible due to being incompatible with the Message Content.

NOTE Based on this definition users must use the 'ALLOW-ALL' expression if you want to define expressions that filter XML content but not JSON content, for example.

For expression types that use namespaces (such as XPath), multiple namespace prefixes and names are added upon session creation.

NOTE An empty result from an expression evaluation will result in the whole message being is filtered; the message content itself is not filtered.

NOTE When expressions are present, a message will only be visible to the receiving application and/or have a notification generated for it if all applicable expressions match both the content type and evaluate to a non-empty result on the content.

NOTE Alternative expressions can be provided by opening multiple sessions on the same channel, each with its own filter expression.

4.5 Message Expiry

During posting of certain messages, a sender MAY specify an expiry duration for the message. An ISBM Service Provider MUST not deliver an expired message to potential receivers unless the receiver has already read the message. If the message was read, then it MUST remain visible to that particular receiver. This is to ensure the message is always available to the receiver so that message removal removes the correct message.

If a sender specifies a negative Expiry duration, then an ISBM Service Provider MUST consider it equivalent to a blank duration.

NOTE Responses can still be posted for a previously read expired request message because the receiver has no indication that the message expired, and Consumers will may still receive response notifications and be able to read and remove these responses.

4.6 Feature Set Declaration

All ISBM Service Providers MUST declare their supported feature set through the ISBM Configuration Discovery Service (see Section 5.9).

The ISBM Configuration Discovery Service allows an ISBM Service Provider to provide information regarding its supported feature set in a machine interpretable way, allowing clients to configure themselves appropriately. The declared features are for the specific instance of the provider, not the possible capabilities of the implementation; those should be documented by the supplier.

Features that MUST be declared by an ISBM Service Provider include:

- Security level conformance
- Supported authentication token types
- Whether content-based filtering is supported
- Supported expression languages/versions for content-based filtering

The ISBM Configuration Discovery Service is limited to the configuration of the service provider itself and not the applications that use it. The discovery of applications, their capabilities, and their configurations (i.e., channels and topics) will be defined in future specifications.

5 Service Definitions

All services defined in ISA 95.00.06 are defined as SOAP Web Services or REST services in this specification. The SOAP and REST service definitions below are to be interpreted in the context of the corresponding ISA 95.00.06 service.

NOTE ISA 95.00.06 does not define an Expire Request operation within the Consumer Request Service, but it has been specified below for a consistent message expiry model across services.

NOTE ISA 95.00.06 does not define a Configuration Discovery Service for the MSM, but it has been specified below to allow an ISBM service provider to dynamically report to applications the supported feature set of the implementation. For example, the languages supported by content-based filtering.

All service operations (except Configuration Discovery Service) have corresponding HTTP examples for SOAP interface shown in Example HTTP Flows. The examples for the Configuration Discovery Service and the REST interface for all the service operations will be provided in the future revisions of the specification.

5.1 Conformance to ISA 95.00.06

The following Service Definitions have been assessed for conformance to the requirements specified in ISA 95.00.06 according to the provisions therein and listed as follows:

Terminology:

- All terms defined in ISA 95.00.06 have been used as such in this specification. Where a defined term is used as a schema element, the term is written in CamelCase notation to clearly identify it as such. Refer to Section 3.1.

Services:

- Channel Management Services
    - Create Channel Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Add Security Tokens Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Partial compliance
            - In addition to the ChannelFault defined in ISA 95.00.06, OperationFault is returned if a SecurityToken is being added to a Channel that was created without any security tokens.
    - Remove Security Tokens Service
        - Notification service support – N/A

        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Delete Channel Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Get Channel Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Get Channels Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
- Notification Service
    - Notify Listener Service
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
            - ListenerURL is the implementation technology specific Listener Identification for both SOAP and REST interface implementations.
- Expiration Listener Service (proposed change)
    - Message Expired Service
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
            - ExpirationListenerURL is the implementation technology specific Expiration Listener Identification for both SOAP and REST interface implementations
- Provider Publication Services
    - Open Publication Session Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Post Publication Service

        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Expire Publication Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Close Publication Session Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
- Consumer Publication Services
    - Open Subscription Session Service
        - Notification service support – Specification Compliant (optional capability), actual level of compliance is implementation specific
        - Filter expression support – Full support in specification, actual level of support is implementation specific
        - Definition of service and optional elements – Compliant
    - Read Publication Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Remove Publication Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Close Subscription Session Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
- Provider Request Services
    - Open Provider Request Service

        - Notification service support – Specification Compliant (optional capability), actual level of compliance is implementation specific
        - Filter expression support – Full support in specification, actual level of support is implementation specific
        - Definition of service and optional elements – Compliant
    - Read Request Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Remove Request Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Post Response Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Close Provider Request Session Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
- Consumer Request Services
    - Open Consumer Request Service
        - Notification service support – Specification Compliant (optional capability), actual level of compliance is implementation specific
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Post Request Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Expire Request Service (proposed change)
        - Notification service support – N/A

        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Read Response Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Remove Response Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant
    - Close Consumer Request Session Service
        - Notification service support – N/A
        - Filter expression support – N/A
        - Definition of service and optional elements – Compliant

All the services listed in ISA-95.00.06 requiring a SecurityToken as input for authentication do not have the SecurityToken listed as an explicit input in the definitions of this specification. The services are still considered compliant with ISA-96.00.06, however, as the SecurityToken has been removed from the input to be defined in implementation specific ways for each implementation technology. That is, the SecurityToken is provided via the WS-Security headers for the SOAP Web Services and provided via the HTTP/1.1 authentication and authorization headers for the REST interface.

Any implementation of the ISBM specification that is assessed as conforming to the ISBM specification will be considered to conform to ISA 95.00.06 as well, with the exceptions noted above.

NOTE While the ISBM specification supports all optional elements of IS-95.00.06, the actual level of support for optional elements is determined by each ISBM Service Provider implementation.

5.2 Channel Management Service

The Channel Management Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML.

5.2.1 Create Channel

The Create Channel service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	CreateChannel
Description	Creates a new channel.
Input	ChannelURI [1] ChannelType [1] ChannelDescription [0..1]

	SecurityToken [0..*]
Behavior	If the ChannelURI already exists, then a ChannelFault is returned. The SecurityTokens are assigned to the channel upon its creation. If duplicate SecurityTokens exist, these result in a single token being assigned to the channel to maintain a distinct list.
Output	N/A
Faults	ChannelFault

5.2.1.1 SOAP Interface

The Create Channel general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**CreateChannel**
Input	CreateChannel (isbm:CreateChannel) – ChannelURI (xs:string) [1] – ChannelType (isbm:ChannelType) [1] – ChannelDescription (xs:string) [0..1] – SecurityToken (isbm:SecurityToken) [0..*]
Output	CreateChannelResponse (isbm:CreateChannelResponse) – No content
Faults	ChannelFault (isbm:ChannelFault)

5.2.1.2 REST Interface

The Create Channel general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**CreateChannel**
HTTP Method	POST
URL	/channels
HTTP Body	Channel (json:Channel) [1] – ChannelURI “uri” (json:string) [1] – ChannelType “channelType” (json:ChannelType) [1] – ChannelDescription “description” (json:string) [0..1]

	– SecurityToken “securityTokens” (json:SecurityToken) [0..*]
HTTP Response (Success)	201 Created
Output	Channel (json:Channel) [1], composed as above excluding “securityTokens”
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 409 Conflict

NOTE Although not required by the general interface, success returns the Channel object in conformance with the requirements of a HTTP 201 response [https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html].

NOTE The output Channel omits the SecurityTokens to prevent leakage of sensitive information.

5.2.2 Add Security Tokens

The Add Security Tokens service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	**AddSecurityTokens**
Description	Adds security tokens to a channel.
Input	ChannelURI [1] SecurityToken [1..*]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned. If a specified SecurityToken is already assigned to the channel, then no further action is taken to maintain a distinct list. If a SecurityToken is being added to a Channel that was currently has no security tokens, then an OperationFault MUST be returned.
Output	N/A
Faults	ChannelFault OperationFault – This is in addition to the Service Definition provided by ISA 95.00.06

NOTE The addition of the OperationFault is a security measure to prevent a malicious client blocking access to channels that are supposed to be “open”. It may also be desirable to prevent deletion of such channels except via authenticated or whitelisted clients.

5.2.2.1 SOAP Interface

The Add Security Tokens general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**AddSecurityTokens**
Input	AddSecurityTokens (isbm:AddSecurityTokens) – ChannelURI (xs:string) [1] – SecurityToken (isbm:SecurityToken) [1..*]

Output	AddSecurityTokensResponse (isbm:AddSecurityTokensResponse) – No content
Faults	ChannelFault (isbm:ChannelFault) OperationFault (isbm:OperationFault)

5.2.2.2 REST Interface

The Add Security Tokens general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**AddSecurityTokens**
HTTP Method	POST
URL	/channels/{channel-uri}/security-tokens
HTTP Body	addSecurityTokens (json:addSecurityTokens) – SecurityToken “securityTokens” (json:SecurityToken) [1..*]
HTTP Response (Success)	201 Created
Output	N/A
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found OperationFault (json:OperationFault) – 409 Conflict

NOTE Returns nothing in contrast to the requirements of a HTTP 201 response [https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html], which would expect the security tokens to be returned, to prevent leakage of sensitive information.

5.2.3 Remove Security Tokens

The Remove Security Tokens service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	**RemoveSecurityTokens**
Description	Removes security tokens from a channel.
Input	ChannelURI [1] SecurityToken [1..*]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned. If any specified SecurityToken is not assigned to the channel, then a SecurityTokenFault is returned. No tokens are removed from the channel, even if they are valid.
Output	N/A

Faults	ChannelFault SecurityTokenFault

5.2.3.1 SOAP Interface

The Remove Security Tokens general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	RemoveSecurityTokens
Input	RemoveSecurityTokens (isbm:RemoveSecurityTokens) – ChannelURI (xs:string) [1] – SecurityToken (isbm:SecurityToken) [1..*]
Output	RemoveSecurityTokensResponse (isbm:RemoveSecurityTokensResponse) – No content
Faults	ChannelFault (isbm:ChannelFault) SecurityTokenFault (isbm:SecurityTokenFault)

5.2.3.2 REST Interface

The Remove Security Tokens general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	RemoveSecurityTokens
HTTP Method	DELETE
URL	/channels/{channel-uri}/security-tokens
HTTP Body	removeSecurityTokens (json:removeSecurityTokens) – SecurityToken “securityTokens” (json:SecurityToken) [1..*]
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found SecurityTokenFault (json:SecurityTokenFault) – 409 Conflict

5.2.4 Delete Channel

The Delete Channel service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	DeleteChannel
Description	Deletes a channel.
Input	ChannelURI [1]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned. The channel along with associated sessions and messages are deleted. No notification is provided to any applications with active sessions.
Output	N/A
Faults	ChannelFault

5.2.4.1 SOAP Interface

The Delete Channel general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	DeleteChannel
Input	DeleteChannel (isbm:DeleteChannel) – ChannelURI (xs:string) [1]
Output	DeleteChannelResponse (isbm:DeleteChannelResponse) – No content
Faults	ChannelFault (isbm:ChannelFault)

5.2.4.2 REST Interface

The Delete Channel general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	DeleteChannel
HTTP Method	DELETE
URL	/channels/{channel-uri}
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found

5.2.5 Get Channel

The Get Channel service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	GetChannel
Description	Gets information about a channel.
Input	ChannelURI [1]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned.
Output	Channel [1], composed of: ChannelURI [1] ChannelType [1] ChannelDescription [0..1]
Faults	ChannelFault

5.2.5.1 SOAP Interface

The Get Channel general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	GetChannel
Input	GetChannel (isbm:GetChannel) – ChannelURI (xs:string) [1]
Output	GetChannelResponse (isbm:GetChannelResponse) – Channel (isbm:Channel) [1], composed of: o ChannelURI (xs:string) [1] o ChannelType (isbm:ChannelType) [1] o ChannelDescription (xs:string) [0..1]
Faults	ChannelFault (isbm:ChannelFault)

5.2.5.2 REST Interface

The Get Channel general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	GetChannel
HTTP Method	GET

URL	/channels/{channel-uri}
HTTP Body	N/A
HTTP Response (Success)	200 OK
Output	Channel (json:Channel) [1], composed of: – ChannelURI “uri” (json:string) [1] – ChannelType “channelType” (json:ChannelType) [1] – ChannelDescription “description” (json:string) [0..1]
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found

NOTE The output Channel omits the SecurityTokens to prevent leakage of sensitive information.

5.2.6 Get Channels

The Get Channels service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	GetChannels
Description	Gets information about all channels.
Input	N/A
Behavior	The channels returned are filtered by those that match the security token. Any channel that does not have security tokens assigned are returned regardless.
Output	Channel [0..*], composed of: ChannelURI [1] ChannelType [1] ChannelDescription [0..1]
Faults	N/A

5.2.6.1 SOAP Interface

The Get Channels general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	GetChannels
Input	N/A
Output	GetChannelsResponse (isbm:GetChannelsResponse) – Channel (isbm:Channel) [0..*], composed of: o ChannelURI (xs:string) [1]

	○ ChannelType (isbm:ChannelType) [1] ○ ChannelDescription (xs:string) [0..1]
Faults	N/A

5.2.6.2 REST Interface

The Get Channels general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**GetChannels**
HTTP Method	GET
URL	/channels
HTTP Body	N/A
HTTP Response (Success)	200 OK
Output	Channel (json:Channel) [0..*], composed of: – ChannelURI “uri” (json:string) [1] – ChannelType “channelType” (json:ChannelType) [1] – ChannelDescription “description” (json:string) [0..1]
HTTP Response (Error)	N/A

NOTE The output Channel(s) omits the SecurityTokens to prevent leakage of sensitive information.

5.3 Notification Service

The Notification Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML. The notification service is a callback from the ISBM provider to an application that has opened a channel using a notification option. The NotifyListener service is a service that is to be implemented by the application. The notification service provides the ability to wait for responses and not require polling of sessions to determine if a message is available.

5.3.1 Notify Listener

The Notify Listener service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	**NotifyListener**
Description	Provides a notification of a new message being able to be read for a session. The Listener URL invoked was provided when the application desiring notifications subscribed to the channel.
Input	SessionID [1]

	MessageID [1] Topic [0..*] RequestMessageID [0..1]
Behavior	Topic MUST NOT be used for consumer request session response notification. RequestMessageID allows correlation with the original request and thus it MUST only be used for consumer request session response notification.
Output	N/A
Faults	N/A

5.3.1.1 SOAP Interface

The Notify Listener general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**NotifyListener**
Input	NotifyListener (isbm:NotifyListener) – SessionID (xs:string) [1] – MessageID (xs:string) [1] – Topic (xs:string) [0..*] – RequestMessageID (xs:string) [0..1]
Output	NotifyListenerResponse (isbm:NotifyListenerResponse) – No content
Faults	N/A

5.3.1.2 REST Interface

The Notify Listener general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**NotifyListener**
HTTP Method	PUT
URL	/notifications/{session-id}/{message-id}
HTTP Body	Notification (json:Notification) [1], composed of: – Topic “topics” (json:string) [0..*] – RequestMessageID “requestMessageId” (json:string) [0..1]

HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	N/A

NOTE SessionID and MessageID are provided in the URL to identify the resource for the PUT method. Both SessionID “session-id” and MessageID “message-id” have been used to ensure uniqueness across different sessions. It is not necessary to include them in the HTTP body.

5.4 Expiration Listener Service

The Expiration Listener Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML. The expiration listener service is a callback from the ISBM provider to an application that has opened a session on a channel using the expiration option. The MessageExpired service is a service that is to be implemented by the application. The expiration listener service provides the ability to proactively act on explicit message expiries, such as cancelling the processing of an in-progress message or forwarding the expiration onto another channel or queue.

5.4.1 Message Expired

The Message Expired service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	MessageExpired
Description	Provides a notification of a publication or request message being explicitly expired by its source. The Expiration Listener URL invoked was provided when the application desiring expiration notifications subscribed to the channel.
Input	SessionID [1] MessageID [1] OriginalMessageID [0..1]
Behavior	OriginalMessageID allows correlation with a message forwarded between channels and thus it MUST be provided if the message was posted with an OriginalMessageID.
Output	N/A
Faults	N/A

5.4.1.1 SOAP Interface

The Message Expired general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	MessageExpired
Input	MessageExpired (isbm:MessageExpired) – SessionID (xs:string) [1] – MessageID (xs:string) [1]

	– OriginalMessageID (xs:string) [0..1]
Output	MessageExpiredResponse (isbm:MessageExpiredResponse) – No content
Faults	N/A

5.4.1.2 REST Interface

The Message Expired general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**MessageExpired**
HTTP Method	PUT
URL	/expirations/{session-id}/{message-id}
HTTP Body	Expiration (json:Expiration) [1], composed of: – OriginalMessageID “originalMessageId” (json:string) [0..1]
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	N/A

NOTE SessionID and MessageID are provided in the URL to identify the resource for the PUT method. Both SessionID “session-id” and MessageID “message-id” have been used to ensure uniqueness across different sessions. It is not necessary to include them in the HTTP body.

5.5 Provider Publication Service

The Provider Publication Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML.

5.5.1 Open Publication Session

The Open Publication Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	**OpenPublicationSession**
Description	Opens a publication session for a channel.
Input	ChannelURI [1]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned. If the channel type is not a Publication type, then an OperationFault is returned.

Output	SessionID [1]
Faults	ChannelFault OperationFault

5.5.1.1 SOAP Interface

The Open Publication Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	OpenPublicationSession
Input	OpenPublicationSession (isbm:OpenPublicationSession) – ChannelURI (xs:string) [1]
Output	OpenPublicationSessionResponse (isbm:OpenPublicationSessionResponse) – SessionID (xs:string) [1]
Faults	ChannelFault (isbm:ChannelFault) OperationFault (isbm:OperationFault)

5.5.1.2 REST Interface

The Open Publication Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	OpenPublicationSession
HTTP Method	POST
URL	/channels/{channel-uri}/publication-sessions
HTTP Body	N/A
HTTP Response (Success)	201 Created
Output	Session (json:Session) [1] – SessionID “sessionId” (json:string) [1]
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found OperationFault (json:OperationFault) – 422 Unprocessable Entity

NOTE No body is required as the ChannelURI is provided via the URL and no other input fields are required.

5.5.2 Post Publication

The Post Publication service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	PostPublication
Description	Posts a publication message on a channel.
Input	SessionID [1] MessageContent [1] Topic [1..*] Expiry [0..1] OriginalMessageID [0..1]
Behavior	If the SessionID does not exist or does not correspond to a publication session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. If a received message is being forwarded to another channel and the received message has an OriginalMessageID, then the OriginalMessageID of the field of the Post Publication MUST be provided and match that of the received message; otherwise, the MessageID of the received message MUST be provided as the OriginalMessageID of the Post Publication.
Output	MessageID [1]
Faults	SessionFault

NOTE OriginalMessageID can be used by the Read Publication receivers to filter out possible duplicated messages.

5.5.2.1 SOAP Interface

The Post Publication general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	PostPublication
Input	PostPublication (isbm:PostPublication) – SessionID (xs:string) [1] – MessageContent (isbm:MessageContent) [1] – Topic (xs:string) [1..*] – Expiry (xs:duration) [0..1] – OriginalMessageID (xs:string) [0..1]
Output	PostPublicationResponse (isbm:PostPublicationResponse) – MessageID (xs:string) [1]
Faults	SessionFault (isbm:SessionFault)

5.5.2.2 REST Interface

The Post Publication general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	PostPublication
HTTP Method	POST
URL	/sessions/{session-id}/publications
HTTP Body	Message (json:Message) [1], composed of: – MessageContent “messageContent” (json:MessageContent) [1] – Topic “topics” (json:string) [1..*] – Expiry “expiry” (json:string) [0..1] – OriginalMessageID “originalMessageId” (json:string) [0..1]
HTTP Response (Success)	201 Created
Output	Message (json:Message) [1], composed of: – MessageID “messageId” (json:string) [1]
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a publication session type

NOTE Only MessageID “messageId” must be returned for the newly created Message to avoid unnecessary resource usage, particularly when the MessageContent is large.

5.5.3 Expire Publication

The Expire Publication service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	ExpirePublication
Description	Expires a posted publication.
Input	SessionID [1] MessageID [1]
Behavior	If the SessionID does not exist or does not correspond to a publication session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. If the MessageID does not correspond with the SessionID or the corresponding message has already expired, then no further action is taken. The message is expired for all topics associated with the message. If any Subscription Sessions have been opened with an Expiration Listener URL, then the Expiration Listener service at each registered URL will be called with the message details.

Output	N/A
Faults	SessionFault

5.5.3.1 SOAP Interface

The Expire Publication general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	ExpirePublication
Input	ExpirePublication (isbm:ExpirePublication) – SessionID (xs:string) [1] – MessageID (xs:string) [1]
Output	ExpirePublicationResponse (isbm:ExpirePublicationResponse) – No content
Faults	SessionFault (isbm:SessionFault)

5.5.3.2 REST Interface

The Expire Publication general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	ExpirePublication
HTTP Method	DELETE
URL	/sessions/{session-id}/publications/{message-id}
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a publication session type

5.5.4 Close Publication Session

The Close Publication Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	ClosePublicationSession
Description	Closes a publication session.
Input	SessionID [1]

Behavior	If the SessionID does not exist (non-existent or already closed) or does not correspond to a publication session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. All unexpired messages that have been posted during the session will be expired.
Output	N/A
Faults	SessionFault

5.5.4.1 SOAP Interface

The Close Publication Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**ClosePublicationSession**
Input	ClosePublicationSession (isbm:ClosePublicationSession) – SessionID (xs:string) [1]
Output	ClosePublicationSessionResponse (isbm:ClosePublicationSessionResponse) – No content
Faults	SessionFault (isbm:SessionFault)

5.5.4.2 REST Interface

The Close Publication Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Note This REST Interface is shared by Close Publication Session, Close Subscription Session, NOTE SessionID and RequestMessageID are not required in the body as they are provided via the URL.

NOTE Only MessageID “messageId” must be returned for the newly created Message to avoid unnecessary resource usage, particularly when the MessageContent is large.

Close Provider Request Session, and Close Consumer Request Session.

Name	**ClosePublicationSession**
HTTP Method	DELETE
URL	/sessions/{session-id}
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A

HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found

5.6 Consumer Publication Service

The Consumer Publication Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML.

5.6.1 Open Subscription Session

The Open Subscription Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	OpenSubscriptionSession
Description	Opens a subscription session for a channel.
Input	ChannelURI [1] Topic [1..*] ListenerURL [0..1] FilterExpression [0..*] ExpirationListenerURL [0..1]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned. If the channel type is not a Publication type, then an OperationFault is returned. If multiple NamespacePrefixes exist with different NamespaceNames in the FilterExpression, then a NamespaceFault is returned.
Output	SessionID [1]
Faults	ChannelFault NamespaceFault OperationFault

5.6.1.1 SOAP Interface

The Open Subscription Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	OpenSubscriptionSession
Input	OpenSubscriptionSession (isbm:OpenSubscriptionSession) – ChannelURI (xs:string) [1] – Topic (xs:string) [1..*]

	– ListenerURL (xs:string) [0..1] – FilterExpression (isbm:FilterExpression) [0..*] – ExpirationListenerURL (xs:string) [0..1]
Output	OpenSubscriptionSessionResponse (isbm:OpenSubscriptionSessionResponse) – SessionID (xs:string) [1]
Faults	ChannelFault (isbm:ChannelFault) NamespaceFault (isbm:NamespaceFault) OperationFault (isbm:OperationFault)

5.6.1.2 REST Interface

The Open Subscription Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**OpenSubscriptionSession**
HTTP Method	POST
URL	/channels/{channel-uri}/subscription-sessions
HTTP Body	Session (json:Session) [1], composed of: – Topic “topics” (json:string) [1..*] – ListenerURL “listenerUrl” (json:string) [0..1] – FilterExpression “filterExpressions” (json:FilterExpression) [0..*] – ExpirationListenerURL “expirationListenerUrl” (json:string) [0..1]
HTTP Response (Success)	201 Created
Output	Session (json:Session) [1] – SessionID “sessionId” (json:string) [1]
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found NamespaceFault (json:Namespacefault) – 400 Bad Request OperationFault (json:OperationFault) – 422 Unprocessable Entity

5.6.2 Read Publication

The Read Publication service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	**ReadPublication**
Description	Returns the first non-expired publication message or a previously read expired message that satisfies the session message filters.
Input	SessionID [1]

Behavior	If the SessionID does not exist or does not correspond to a publication session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. The returned Topics will correspond to the intersection of the Topics of the posted publication and the Topics specified in the subscription session. If there was an Original Message ID on the Post Publication, then it is returned in the Original Message ID. If there was no Original Message ID on the Post Publication service, then no Original Message ID field is returned.
Output	PublicationMessage [0..1], composed of: – MessageID [1] – MessageContent [1] – Topic [1..*] – OriginalMessageID [0..1]
Faults	SessionFault

NOTE OriginalMessageID can be used by the Read Publication receivers to filter out possible duplicated messages.

5.6.2.1 SOAP Interface

The Read Publication general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**ReadPublication**
Input	ReadPublication (isbm:ReadPublication) – SessionID (xs:string) [1]
Output	ReadPublicationResponse (isbm:ReadPublicationResponse) – PublicationMessage (isbm:PublicationMessage) [0..1], composed of: o MessageID (xs:string) [1] o MessageContent (isbm:MessageContent) [1] o Topic (xs:string) [1..*] o OriginalMessageID (xs:string) [0..1]
Faults	SessionFault (isbm:SessionFault)

NOTE A no message response is a success with no PublicationMessage element.

5.6.2.2 REST Interface

The Read Publication general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	ReadPublication
HTTP Method	GET
URL	/sessions/{session-id}/publication
HTTP Body	N/A
HTTP Response (Success)	200 OK
Output	Message (json:Message) [1], composed of: – MessageID “messageId” (json:string) [1] – MessageContent “messageContent” (json:MessageContent) [1] – Topic “topics” (json:string) [1..*] – OriginalMessageID “originalMessageId” (json:string) [0..1]
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a subscription session type

NOTE A no message response is returned as a 404 Not Found rather than an "empty" message as it maps better to the concept of resources in a REST API: if there are no messages on the queue, the resource does not exist and, hence, 404 should be returned.

5.6.3 Remove Publication

The Remove Publication service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	RemovePublication
Description	Removes the first, if any, publication message in the subscription queue.
Input	SessionID [1]
Behavior	If the SessionID does not exist or does not correspond to a publication session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned.
Output	N/A
Faults	SessionFault

5.6.3.1 SOAP Interface

The Remove Publication general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	RemovePublication
Input	RemovePublication (isbm:RemovePublication) – SessionID (xs:string) [1]
Output	RemovePublicationResponse (isbm:RemovePublicationResponse) – No Content
Faults	SessionFault (isbm:SessionFault)

5.6.3.2 REST Interface

The Remove Publication general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	RemovePublication
HTTP Method	DELETE
URL	/sessions/{session-id}/publication
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a subscription session type

5.6.4 Close Subscription Session

The Close Subscription Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	CloseSubscriptionSession
Description	Closes a subscription session.
Input	SessionID [1]
Behavior	If the SessionID does not exist (non-existent or already closed) or does not correspond to a publication session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned.
Output	N/A
Faults	SessionFault

5.6.4.1 SOAP Interface

The Close Subscription Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	CloseSubscriptionSession
Input	CloseSubscriptionSession (isbm:CloseSubscriptionSession) – SessionID (xs:string) [1]
Output	CloseSubscriptionSessionResponse (isbm:CloseSubscriptionSessionResponse) – No Content
Faults	SessionFault (isbm:SessionFault)

5.6.4.2 REST Interface

The Close Subscription Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Note This REST Interface is shared by Close Publication Session, Close Subscription Session, NOTE SessionID and RequestMessageID are not required in the body as they are provided via the URL.

NOTE Only MessageID “messageId” must be returned for the newly created Message to avoid unnecessary resource usage, particularly when the MessageContent is large.

Close Provider Request Session, and Close Consumer Request Session.

Name	CloseSubscriptionSession
HTTP Method	DELETE
URL	/sessions/{session-id}
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found

5.7 Provider Request Service

The Provider Request Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML.

5.7.1 Open Provider Request Session

The Open Provider Request Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	OpenProviderRequestSession
Description	Opens a provider request session for a channel for reading requests and posting responses.
Input	ChannelURI [1] Topic [1..*] ListenerURL [0..1] FilterExpression [0..*] ExpirationListenerURL [0..1]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned. If the channel type is not a Request type, then an OperationFault is returned. If multiple NamespacePrefixes exist with different NamespaceNames in the FilterExpression, then a NamespaceFault is returned.
Output	SessionID [1]
Faults	ChannelFault NamespaceFault OperationFault

5.7.1.1 SOAP Interface

The Open Provider Request Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	OpenProviderRequestSession
Input	OpenProviderRequestSession (isbm:OpenProviderRequestSession) – ChannelURI (xs:string) [1] – Topic (xs:string) [1..*] – ListenerURL (xs:string) [0..1] – FilterExpression (isbm:FilterExpression) [0..*] – ExpirationListenerURL (xs:string) [0..1]
Output	OpenProviderRequestSessionResponse (isbm:OpenProviderRequestSessionResponse) – SessionID (xs:string) [1]
Faults	ChannelFault (isbm:ChannelFault)

	NamespaceFault (isbm:NamespaceFault) OperationFault (isbm:OperationFault)

5.7.1.2 REST Interface

The Open Provider Request Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	OpenProviderRequestSession
HTTP Method	POST
URL	/channels/{channel-uri}/provider-request-sessions
HTTP Body	Session (json:Session) [1], composed of: – Topic “topics” (json:string) [1..*] – ListenerURL “listenerUrl” (json:string) [0..1] – FilterExpression “filterExpressions” (json:FilterExpression) [0..*] – ExpirationListenerURL “expirationListenerUrl” (json:string) [0..1]
HTTP Response (Success)	201 Created
Output	Session (json:Session) [1], composed of: – SessionID “sessionId” (json:string) [1]
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found NamespaceFault (json:Namespacefault) – 400 Bad Request OperationFault (json:OperationFault) – 422 Unprocessable Entity

5.7.2 Read Request

The Read Request service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	ReadRequest
Description	Returns the first non-expired request message or a previously read expired message that satisfies the session message filters.
Input	SessionID [1]
Behavior	If the SessionID does not exist or does not correspond to a provider request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. The returned Topic will correspond to the first topic that matched the posted request.

	If there was an Original Message ID on the Post Request, then it is returned in the Original Message ID. If there was no Original Message ID on the Post Request service, then no Original Message ID field is returned.
Output	RequestMessage [0..1], composed of: – MessageID [1] – MessageContent [1] – Topic [1] – OriginalMessageID [0..1]
Faults	SessionFault

NOTE OriginalMessageID can be used by the Read Request receivers to filter out possible duplicated messages.

5.7.2.1 SOAP Interface

The Read Request general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**ReadRequest**
Input	ReadRequest (isbm:ReadRequest) – SessionID (xs:string) [1]
Output	ReadRequestResponse (isbm:ReadRequestResponse) – RequestMessage (isbm:RequestMessage) [0..1], composed of: o MessageID (xs:string) [1] o MessageContent (isbm:MessageContent) [1] o Topic (xs:string) [1] o OriginalMessageID (xs:string) [0..1]
Faults	SessionFault (isbm:SessionFault)

NOTE A no message response is a success with no RequestMessage element.

5.7.2.2 REST Interface

The Read Request general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**ReadRequest**
HTTP Method	GET
URL	/sessions/{session-id}/request
HTTP Body	N/A

HTTP Response (Success)	200 OK
Output	Message (json:Message) [1], composed of: – MessageID “messageId” (json:string) [1] – MessageContent “messageContent” (json:MessageContent) [1] – Topic “topics” (json:string) [1] – OriginalMessageID “originalMessageId” (json:string) [0..1]
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a provider request session type

NOTE A no message response is returned as a 404 Not Found rather than an "empty" message as it maps better to the concept of resources in a REST API: if there are no messages on the queue, the resource does not exist and, hence, 404 should be returned.

NOTE The output Message conforms to the basic schema, hence, the single Topic is represented as a JSON array with a single value: this is guaranteed by the Post Request operation only allowing a single Topic value.

5.7.3 Remove Request

The Remove Request service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	RemoveRequest
Description	Deletes the first request message, if any, in the session message queue.
Input	SessionID [1]
Behavior	If the SessionID does not exist or does not correspond to a provider request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned.
Output	N/A
Faults	SessionFault

5.7.3.1 SOAP Interface

The Remove Request general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	RemoveRequest
Input	RemoveRequest (isbm:RemoveRequest) – SessionID (xs:string) [1]
Output	RemoveRequestResponse (isbm:RemoveRequestResponse) – No Content

Faults	SessionFault (isbm:SessionFault)

5.7.3.2 REST Interface

The Remove Request general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**RemoveRequest**
HTTP Method	DELETE
URL	/sessions/{session-id}/request
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a provider request session type

5.7.4 Post Response

The Post Response service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	**PostResponse**
Description	Posts a response message on a channel.
Input	SessionID [1] RequestMessageID [1] MessageContent [1] OriginalMessageID [0..1]
Behavior	If the SessionID does not exist or does not correspond to a provider request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. If there is no request message that can be matched to RequestMessageID, then no further action is taken. If a received response is being forwarded to another channel and the response message has an OriginalMessageID, then the OriginalMessageID of the field of the Post Response MUST be provided and match that of the received response; otherwise, the MessageID of the received response MUST be provided as the OriginalMessageID of the forwarded Post Response.

Output	MessageID [1]
Faults	SessionFault

NOTE If there is no unexpired request message that can be matched to RequestMessageID, then no further action is taken.

NOTE An implementation may choose to generate a MessageID for the response when no action is taken, or it may output an “empty” MessageID.

NOTE OriginalMessageID can be used by the Read Response receivers to filter out possible duplicated messages.

5.7.4.1 SOAP Interface

The Post Response general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**PostResponse**
Input	PostResponse (isbm:PostResponse) – SessionID (xs:string) [1] – RequestMessageID (xs:string) [1] – MessageContent (isbm:MessageContent) [1] – OriginalMessageID (xs:string) [0..1]
Output	PostResponseResponse (isbm:PostResponseResponse) – MessageID (xs:string) [1]
Faults	SessionFault (isbm:SessionFault)

5.7.4.2 REST Interface

The Post Response general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**PostResponse**
HTTP Method	POST
URL	/sessions/{session-id}/requests/{request-message-id}/responses
HTTP Body	Message (json:Message) [1], composed of: – MessageContent “messageContent” (json:MessageContent) [1] – OriginalMessageId “originalMessageId” (json:string) [0..1]
HTTP Response (Success)	201 Created
Output	Message (json:Message) [1], composed of: – MessageID “messageId” (json:string) [1]

HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a provider request session type

NOTE SessionID and RequestMessageID are not required in the body as they are provided via the URL.

NOTE Only MessageID "messageId" must be returned for the newly created Message to avoid unnecessary resource usage, particularly when the MessageContent is large.

5.7.5 Close Provider Request Session

The Close Provider Request Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	CloseProviderRequestSession
Description	Closes a provider request session.
Input	SessionID [1]
Behavior	If the SessionID does not exist (non-existent or already closed) or does not correspond to a Request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned.
Output	N/A
Faults	SessionFault

5.7.5.1 SOAP Interface

The Close Provider Request Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	CloseProviderRequestSession
Input	CloseProviderRequestSession (isbm:CloseProviderRequestSession) – SessionID (xs:string) [1]
Output	CloseProviderRequestSessionResponse (isbm:CloseProviderRequestSessionResponse) – No Content
Faults	SessionFault (isbm:SessionFault)

5.7.5.2 REST Interface

The Close Provider Request Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Note This REST Interface is shared by Close Publication Session, Close Subscription Session, NOTE SessionID and RequestMessageID are not required in the body as they are provided via the URL.

NOTE Only MessageID “messageId” must be returned for the newly created Message to avoid unnecessary resource usage, particularly when the MessageContent is large.

Close Provider Request Session, and Close Consumer Request Session.

Name	CloseProviderRequestSession
HTTP Method	DELETE
URL	/sessions/{session-id}
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found

5.8 Consumer Request Service

The Consumer Request Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML.

5.8.1 Open Consumer Request Session

The Open Customer Request Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	OpenConsumerRequestSession
Description	Opens a consumer request session for a channel for posting requests and reading responses.
Input	ChannelURI [1] ListenerURL [0..1]
Behavior	If the ChannelURI does not exist, then a ChannelFault is returned. If the specified channel is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a ChannelFault is returned. If the channel type is not a Request type, then an OperationFault is returned.
Output	SessionID [1]
Faults	ChannelFault OperationFault

5.8.1.1 SOAP Interface

The Open Consumer Request Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	OpenConsumerRequestSession
Input	OpenConsumerRequestSession (isbm:OpenConsumerRequestSession) – ChannelURI (xs:string) [1] – ListenerURL (xs:string) [0..1]
Output	OpenConsumerRequestSessionResponse (isbm:OpenConsumerRequestSessionResponse) – SessionID (xs:string) [1]
Faults	ChannelFault (isbm:ChannelFault) OperationFault (isbm:OperationFault)

5.8.1.2 REST Interface

The Open Consumer Request Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	OpenConsumerRequestSession
HTTP Method	POST
URL	/channels/{channel-uri}/consumer-request-sessions
HTTP Body	Session (json:Session) [1], composed of: – ListenerURL “listenerUrl” (json:string) [0..1]
HTTP Response (Success)	201 Created
Output	Session (json:Session) [1], composed of: – SessionID “sessionId” (json:string) [1]
HTTP Response (Error)	ChannelFault (json:ChannelFault) – 404 Not Found OperationFault (json:OperationFault) – 422 Unprocessable Entity

5.8.2 Post Request

The Post Request service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	PostRequest
Description	Posts a request message on a channel.
Input	SessionID [1] MessageContent [1] Topic [1] Expiry [0..1]

	OriginalMessageID [0..1]
Behavior	If the SessionID does not exist or does not correspond to a consumer request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. If a received request is being forwarded to another channel and the request message has an OriginalMessageID, then the OriginalMessageID of the field of the Post Request MUST be provided and match that of the received request; otherwise, the MessageID of the received request MUST be provided as the OriginalMessageID of the forwarded Post Request.
Output	MessageID [1]
Faults	SessionFault

NOTE OriginalMessageID can be used by the Read Request receivers to filter out possible duplicated messages.

5.8.2.1 SOAP Interface

The Post Request general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**PostRequest**
Input	PostRequest (isbm:PostRequest) – SessionID (xs:string) [1] – MessageContent (isbm:MessageContent) [1] – Topic (xs:string) [1] – Expiry (xs:duration) [0..1] – OriginalMessageID (xs:string) [0..1]
Output	PostRequestResponse (isbm:PostRequestResponse) – MessageID (xs:string) [1]
Faults	SessionFault (isbm:SessionFault)

5.8.2.2 REST Interface

The Post Request general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**PostRequest**
HTTP Method	POST
URL	/sessions/{session-id}/requests

HTTP Body	Message (json:Message) [1], composed of: – MessageContent “messageContent” (json:MessageContent) [1] – Topic “topics” (json:string) [1] – Expiry “expiry” (json:string) [0..1] – OriginalMessageID “originalMessageId” (json:string) [0..1]
HTTP Response (Success)	201 Created
Output	Message (json:Message) [1], composed of: – MessageID “messageId” (json:string) [1]
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a consumer request session type

NOTE The input Message conforms to the basic schema, hence, the single Topic is represented as a JSON array with a single value: any empty array or more than 1 value will output a Parameter Fault.

NOTE Only MessageID “messageId” must be returned for the newly created Message to avoid unnecessary resource usage, particularly when the MessageContent is large.

5.8.3 Expire Request

The Expire Request service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	ExpireRequest
Description	Expires a posted request message.
Input	SessionID [1] MessageID [1]
Behavior	If the SessionID does not exist or does not correspond to a consumer request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. If the MessageID does not correspond with the SessionID or the corresponding message has already expired, then no further action is taken. Any unread responses associated with the request MAY be removed from the queue of the consumer. If any Provider Request Sessions have been opened with an Expiration Listener URL, then the Expiration Listener service at each registerd URL will be called with the message details.
Output	N/A
Faults	SessionFault

NOTE It has been left open for vendor's implementation to document what should happen to responses that have already been posted against a request that has subsequently expired. The two options are, should responses be available for consumer to read or should they be removed from the consumer's queue (unless they have been read previously).

5.8.3.1 SOAP Interface

The Expire Request general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	ExpireRequest
Input	ExpireRequest (isbm:ExpireRequest) – SessionID (xs:string) [1] – MessageID (xs:string) [1]
Output	ExpireRequestResponse (isbm:ExpireRequestResponse) – No content
Faults	SessionFault (isbm:SessionFault)

5.8.3.2 REST Interface

The Expire Request general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	ExpireRequest
HTTP Method	DELETE
URL	/sessions/{session-id}/requests/{message-id}
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a consumer request session type

5.8.4 Read Response

The Read Response service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	ReadResponse
Description	Returns the first response message, if any, in the session message queue associated with the request.

Input	SessionID [1] RequestMessageID [1]
Behavior	If the SessionID does not exist or does not correspond to a consumer request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. If the RequestMessageID does not correspond to a message in the message queue, then no message is returned. If there was an Original Message ID on the Post Response, then it is returned in the Original Message ID. If there was no Original Message ID on the Post Response service, then no Original Message ID field is returned.
Output	ResponseMessage [0..1], composed of: – MessageID [1] – MessageContent [1] – OriginalMessageID [0..1]
Faults	SessionFault

NOTE OriginalMessageID can be used by the Read Response receivers to filter out possible duplicated messages.

5.8.4.1 SOAP Interface

The Read Response general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**ReadResponse**
Input	ReadResponse (isbm:ReadResponse) – SessionID (xs:string) [1] – RequestMessageID (xs:string) [1]
Output	ReadResponseResponse (isbm:ReadResponseResponse) – ResponseMessage (isbm:ResponseMessage) [0..1], composed of: o MessageID (xs:string) [1] o MessageContent (isbm:MessageContent) [1] o OriginalMessageID (xs:string) [0..1]
Faults	SessionFault (isbm:SessionFault)

NOTE A no message response is a success with no ResponseMessage element.

5.8.4.2 REST Interface

The Read Response general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	ReadResponse
HTTP Method	GET
URL	/sessions/{session-id}/requests/{request-message-id}/response
HTTP Body	N/A
HTTP Response (Success)	200 OK
Output	Message (json:Message) [1], composed of: – MessageID “messageId” (json:string) [1] – MessageContent “messageContent” (json:MessageContent) [1] – OriginalMessageID “originalMessageId” (json:string) [0..1]
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a consumer request session type

NOTE A no message response is returned as a 404 Not Found rather than an "empty" message as it maps better to the concept of resources in a RESTful API. If there are no messages on the queue, the resource does not exist and, hence, 404 should be returned.

5.8.5 Remove Response

The Remove Response service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	RemoveResponse
Description	Deletes the first response message, if any, in the session message queue associated with the request.
Input	SessionID [1] RequestMessageID [1]
Behavior	If the SessionID does not exist or does not correspond to a consumer request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. If the RequestMessageID does not correspond to a message in the message queue, then no further action is taken.

Output	N/A
Faults	SessionFault

5.8.5.1 SOAP Interface

The Remove Response general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	RemoveResponse
Input	RemoveResponse (isbm:RemoveResponse) – SessionID (xs:string) [1] – RequestMessageID (xs:string) [1]
Output	RemoveResponseResponse (isbm:RemoveResponseResponse) – No Content
Faults	SessionFault (isbm:SessionFault)

5.8.5.2 REST Interface

The Remove Response general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	RemoveResponse
HTTP Method	DELETE
URL	/sessions/{session-id}/requests/{request-message-id}/response
HTTP Body	N/A
HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found SessionFault (json:SessionFault) – 422 Unprocessable Entity – session exists but does not correspond to a consumer request session type

5.8.6 Close Consumer Request Session

The Close Consumer Request Session service in general MUST have the behavior, inputs, outputs and return the faults as defined by the following table.

Name	CloseConsumerRequestSession
Description	Closes a consumer request session.
Input	SessionID [1]

Behavior	If the SessionID does not exist (non-existent or already closed) or does not correspond to a Request session, then a SessionFault is returned. If the channel associated with the specified session is assigned SecurityToken and the token provided to the operation for authentication does not match a token assigned to the channel, then a SessionFault is returned. All unexpired requests that have been posted during the session will be expired. If any Provider Request Sessions have been opened with an Expiration Listener URL, then the Expiration Listener service at each registered URL will be called with the message details for each request that is expired due to session close.
Output	N/A
Faults	SessionFault

5.8.6.1 SOAP Interface

The Close Consumer Request Session general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**CloseConsumerRequestSession**
Input	CloseConsumerRequestSession (isbm:CloseConsumerRequestSession) – SessionID (xs:string) [1]
Output	CloseConsumerRequestSessionResponse (isbm:CloseConsumerRequestSessionResponse) – No Content
Faults	SessionFault (isbm:SessionFault)

5.8.6.2 REST Interface

The Close Consumer Request Session general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Note This REST Interface is shared by Close Publication Session, Close Subscription Session, NOTE SessionID and RequestMessageID are not required in the body as they are provided via the URL.

NOTE Only MessageID “messageId” must be returned for the newly created Message to avoid unnecessary resource usage, particularly when the MessageContent is large.

Close Provider Request Session, and Close Consumer Request Session.

Name	**CloseConsumerRequestSession**
HTTP Method	DELETE
URL	/sessions/{session-id}
HTTP Body	N/A

HTTP Response (Success)	204 No Content
Output	N/A
HTTP Response (Error)	SessionFault (json:SessionFault) – 404 Not Found

5.9 ISBM Configuration Discovery Service

The ISBM Configuration Discovery Service for SOAP Interface is available as a WSDL description and for REST Interface is available as OpenAPI 3.0.1 descriptions in YAML.

5.9.1 Get Supported Operations

The Get Supported Operations in general MUST have the behavior, inputs, outputs, and return the faults as defined by the following table.

Name	**GetSupportedOperations**
Description	Gets information about the supported operations and features of the ISBM service provider. The purpose of this operation is to allow an application to be configured appropriately to communicate successfully with the service provider.
Input	N/A
Behavior	If the service supports content-based filtering of XML messages, IsXMLFilteringEnabled is set True. If the service supports content-based filtering of JSON messages, IsJSONFilteringEnabled is set True. If content-based filtering is supported, then the value of SupportedContentFilteringLanguages MUST list the supported languages and the applicable MediaTypes and optionally the language versions. Except for the requirements in Section 4.4, the list of supported languages is implementation specific. The authentication mechanisms and token types supported by the service provider is listed in SupportedAuthentications, which provides list of token schemas supported by the SOAP interface and the list of HTTP authentication schemes supported by the REST interface. The scheme names provided in RestSupportedAuthenticationSchemes MUST match one of the scheme names registered in the IANA's HTTP Authentication Scheme Registry [https://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml]. Except for the requirements in Section 4.2, the lists of authentication mechanisms are implementation specific. The security level (refer to Section 7.20) to which the ISBM service provider conforms, is provided by SecurityLevelConformance. If the service supports dead lettering (posting a response to an expired request), IsDeadLetteringEnabled is set True. If the service permits connecting applications to create channels with no security tokens, IsChannelCreationEnabled is set True. If the service permits connecting applications to add security tokens to channels that havenone, IsOpenChannelSecuringEnabled is set True. If the service requires connecting applications to reside in a whitelist, IsWhitelistRequired is set True.

	The expiry duration applicable to all messages that do not have their own individual expiry given is provided by DefaultExpiryDuration. If the DefaultExpiryDuration is provided, applications should not expect any message to exceed this default duration. A negative, null, or empty (may differ by interface type) duration is no duration. The service returns a URL of a human readable webpage containing specific implementation details intended for developers (e.g. configuration and setup information, contact details, help documentation, current status, etc.) in the string AdditionalInformationURL.
Output	SupportedOperations [1], composed of: – IsXMLFilteringEnabled [1] – IsJSONFilteringEnabled [1] – SupportedContentFilteringLanguages [1] o ContentFilteringLanguage [0..*] – SupportedAuthentications [1] o SoapSupportedTokenSchema [0..*] o RestSupportedAuthenticationScheme [0..*] – SecurityLevelConformance [1] – IsDeadLetteringEnabled [1] – IsChannelCreationEnabled [1] – IsOpenChannelSecuringEnabled [1] – IsWhitelistRequired [1] – DefaultExpiryDuration [1] – AdditionalInformationURL [1]
Faults	N/A

NOTE Due to the requirements of Section 4.2, if an ISBM provider implements a specific interface type, e.g., SOAP or REST, then the respective SupportedAuthentications list will have at least 1 item.

5.9.1.1 SOAP Interface

The Get Supported Operations general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**GetSupportedOperations**
Input	GetSupportedOperations (isbm:GetSupportedOperations) - No Content
Output	GetSupportedOperationsResponse (isbm:GetSupportedOperationsResponse) – SupportedOperations (isbm:SupportedOperations) [1], composed of: o IsXMLFilteringEnabled (xs:boolean) [1]

	○ IsJSONFilteringEnabled (xs:boolean) [1] ○ SupportedContentFilteringLanguages (isbm:SupportedContentFilteringLanguages) [1] ▪ ContentFilteringLanguage (isbm:ContentFilteringLanguage) [0..*] ○ SupportedAuthentications [1] ▪ SoapSupportedTokenSchema (isbm:TokenSchema) [0..*] ▪ RestSupportedAuthenticationScheme (isbm:AuthenticationScheme) [0..*] ○ SecurityLevelConformance (isbm:SecurityLevel) [1] ○ IsDeadLetteringEnabled (xs:boolean) [1] ○ IsChannelCreationEnabled (xs:boolean) [1] ○ IsOpenChannelSecuringEnabled (xs:boolean) [1] ○ IsWhitelistRequired (xs:boolean) [1] ○ DefaultExpiryDuration (xs:duration) [1] ○ AdditionalInformationURL (xs:string) [1]
Faults	N/A

5.9.1.2 REST Interface

The Get Supported Operations general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**GetSupportedOperations**
HTTP Method	GET
URL	/configuration/supported-operations
HTTP Body	None
HTTP Response (Success)	200 OK
Output	SupportedOperations (json:SupportedOperations) [1], composed of: – “isXMLFilteringEnabled” (json:boolean) [1] – “isJSONFilteringEnabled” (json:boolean) [1] – “supportedContentFilteringLanguages” (json:SupportedContentFilteringLanguages) [1] ○ “contentFilteringLanguages” (json:ContentFilteringLanguage) [0..*] – “supportedAuthentications” [1] ○ “soapSupportedTokenSchema” (json:TokenSchema) [0..*]

	○ “restSupportedAuthenticationScheme” (json:AuthenticationScheme) [0..*] – “securityLevelConformance” (json:SecurityLevel) [1] – “isDeadLetteringEnabled” (json:boolean) [1] – “isChannelCreationEnabled” (json:boolean) [1] – “isOpenChannelSecuringEnabled” (json:boolean) [1] – “isWhitelistRequired” (json:boolean) [1] – “defaultExpiryDuration” (json:string) [1] – “additionalInformationURL” (json:string) [1]
HTTP Response (Error)	N/A

5.9.2 Get Security Details

The Get Security Details in general MUST have the behavior, inputs, outputs, and return the faults as defined by the following table.

Name	**GetSecurityDetails**
Description	Gets the detailed security related information of the ISBM service provider. The security details are exposed only if the connecting application provides a valid SecurityToken. Each application may be assigned a SecurityToken out-of-band by the service provider.
Input	SecurityToken [1]
Behavior	If the SecurityToken provided to the operation for authentication does not match the assigned token, then a SecurityTokenFault is returned. If the service provides transport layer security (TLS), IsTLSEnabled is set True. If the service requires SecurityTokens to secure all channels, IsSecurityTokenRequired is set True. If the service uses SecurityTokens and the service provider stores SecurityTokens in encrypted format, IsSecurityTokenEncryptionEnabled is set True. If the service requires connecting applications to verify identity with certificates, IsCertificateRequired is set True. If the service provider uses Role-Based Access control (RBAC) for managing configuration and performing operations on the services, IsRBACEnabled is set True. If the service provider uses third party services to encrypt/decrypt security keys and tokens, IsKeyManagementServiceEnabled is set True. If the service performs end-to-end encryption of messages, IsEndToEndMessageEncryptionEnabled is set True.
Output	SecurityDetails [1], composed of: – IsTLSEnabled [1]

	– IsSecurityTokenRequired [1] – IsSecurityTokenEncryptionEnabled [1] – IsCertificateRequired [1] – IsRBACEnabled [1] – IsKeyManagementServiceEnabled [1] – IsEndToEndMessageEncryptionEnabled [1]
Faults	SecurityTokenFault

5.9.2.1 SOAP Interface

The Get Security Details general interface is mapped into SOAP 1.1/1.2 as embedded XML schemas in WSDL descriptions according to the following schema types.

The behavior of the SOAP interface MUST conform to that of the general description.

Name	**GetSecurityDetails**
Input	GetSecurityDetails (isbm:GetSecurityDetails) – No Content
Output	GetSecurityDetailsResponse (isbm:GetSecurityDetailsResponse) – SecurityDetails (isbm:SecurityDetails) [1], composed of: o IsTLSEnabled (xs:boolean) [1] o IsSecurityTokenRequired (xs:boolean) [1] o IsSecurityTokenEncryptionEnabled (xs:boolean) [1] o IsCertificateRequired (xs:boolean) [1] o IsRBACEnabled (xs:boolean) [1] o IsKeyManagementServiceEnabled (xs:boolean) [1] o IsEndToEndMessageEncryptionEnabled (xs:boolean) [1]
Faults	SecurityTokenFault (isbm:SecurityTokenFault)

NOTE The SecurityToken required by the Input of the general interface (Section 5.9.2) is provided using a SOAP specific mechanism and, hence, is not present in the GetSecurityDetails body of the SOAP interface.

5.9.2.2 REST Interface

The Get Security Details general interface is mapped into a RESTful interface as an OpenAPI description according to the following rules.

The behavior of the REST interface MUST conform to that of the general description with necessary adjustments to conform to REST principles and HTTP specifications.

Name	**GetSecurityDetails**
HTTP Method	GET

URL	/configuration/security-details
HTTP Body	None
HTTP Response (Success)	200 OK
Output	SecurityDetails (json:SecurityDetails) [1], composed of: – "isTLSEnabled" (json:boolean) [1] – "isSecurityTokenRequired" (json:boolean) [1] – "isSecurityTokenEncryptionEnabled" (json:boolean) [1] – "isCertificateRequired" (json:boolean) [1] – "isRBACEnabled" (json:boolean) [1] – "isKeyManagementServiceEnabled" (json:boolean) [1] – "isEndToEndMessageEncryptionEnabled" (json:boolean) [1]
HTTP Response (Error)	SecurityTokenFault (json:SecurityTokenFault) – 401 Unauthorized

NOTE The SecurityToken required by the Input of the general interface (Section 5.8.2) is provided using the standard HTTP authentication headers and, hence, is not present in the HTTP Body of the REST interface.

6 XML Data Structures

The following data structures are used by the services defined in Service Definitions and are defined using XML Schema. All types have a target namespace of `http://www.openoandm.org/isbm/`.

6.1 AuthenticationScheme

```
<xs:complexType name="AuthenticationScheme">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="SchemeName">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="SchemeInfoUrl" type="xs:anyURI" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

6.2 Channel

```
<xs:complexType name="Channel">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="ChannelURI" type="xs:string"/>
    <xs:element minOccurs="1" maxOccurs="1" name="ChannelType" type="isbm:ChannelType"/>
    <xs:element minOccurs="0" maxOccurs="1" name="ChannelDescription" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

6.3 ChannelType

```
<xs:simpleType name="ChannelType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Publication"/>
    <xs:enumeration value="Request"/>
  </xs:restriction>
</xs:simpleType>
```

6.4 ContentFilteringLanguage

```
<xs:complexType name="ContentFilteringLanguage">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="languageVersion" type="xs:token" use="optional"/>
      <xs:attribute name="applicableMediaTypes" type="isbm:MediaTypeList" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

6.5 FilterExpression

```
<xs:complexType name="FilterExpression">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="ExpressionString">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
```

```
            <xs:attribute name="language" type="xs:token" use="required"/>
            <xs:attribute name="languageVersion" type="xs:token" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Namespace" type="isbm:Namespace"/>
  </xs:sequence>
  <xs:attribute name="applicableMediaTypes" type="isbm:MediaTypeList" use="optional"/>
</xs:complexType>
```

6.6 MediaTypeList

```
<xs:simpleType name="MediaTypeList">
  <xs:list itemType="xs:token"/>
</xs:simpleType>
```

6.7 MessageContent

```
<xs:complexType name="MessageContent" abstract="true">
</xs:complexType>
<xs:complexType name="BinaryContent">
  <xs:complexContent>
    <xs:extension base="isbm:MessageContent">
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="Content" type="xs:base64Binary" />
      </xs:sequence>
      <xs:attribute use="optional" name="mediaType" type="xs:string" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="StringContent">
  <xs:complexContent>
    <xs:extension base="isbm:MessageContent">
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="Content" type="xs:string" />
      </xs:sequence>
      <xs:attribute use="required" name="mediaType" type="xs:string" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="XMLContent">
  <xs:complexContent>
    <xs:extension base="isbm:MessageContent">
      <xs:sequence>
        <xs:any minOccurs="1" maxOccurs="1" namespace="##any" processContents="lax"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.8 Namespace

```
<xs:complexType name="Namespace">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="NamespacePrefix" type="xs:string"/>
    <xs:element minOccurs="1" maxOccurs="1" name="NamespaceName" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

6.9 PublicationMessage

```
<xs:complexType name="PublicationMessage">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="MessageID" type="xs:string"/>
    <xs:element minOccurs="1" maxOccurs="1" name="MessageContent" type="isbm:MessageContent"/>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="Topic" type="xs:string"/>
    <xs:element minOccurs="0" maxOccurs="1" name="OriginalMessageID" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

6.10 RequestMessage

```
<xs:complexType name="RequestMessage">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="MessageID" type="xs:string"/>
    <xs:element minOccurs="1" maxOccurs="1" name="MessageContent" type="isbm:MessageContent"/>
    <xs:element minOccurs="1" maxOccurs="1" name="Topic" type="xs:string"/>
    <xs:element minOccurs="0" maxOccurs="1" name="OriginalMessageID" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

6.11 ResponseMessage

```
<xs:complexType name="ResponseMessage">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="MessageID" type="xs:string"/>
    <xs:element minOccurs="1" maxOccurs="1" name="MessageContent" type="isbm:MessageContent"/>
    <xs:element minOccurs="0" maxOccurs="1" name="OriginalMessageID" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

6.12 SecurityDetails

```
<xs:complexType name="SecurityDetails">
<xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="IsTLSEnabled" type="xs:boolean"/>
    <xs:element minOccurs="1" maxOccurs="1" name="IsSecurityTokenRequired" type="xs:boolean"/>
    <xs:element minOccurs="1" maxOccurs="1" name="IsSecurityTokenEncryptionEnabled"
type="xs:boolean"/>
    <xs:element minOccurs="1" maxOccurs="1" name="IsCertificateRequired" type="xs:boolean"/>
    <xs:element minOccurs="1" maxOccurs="1" name="IsRBACEnabled" type="xs:boolean"/>
    <xs:element minOccurs="1" maxOccurs="1" name="IsKeyManagementServiceEnabled" type="xs:boolean"/>
    <xs:element minOccurs="1" maxOccurs="1" name="IsEndToEndMessageEncryptionEnabled"
type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

6.13 SecurityLevel

```
<xs:simpleType name="SecurityLevel">
  <xs:restriction base='xs:integer'>
    <xs:minInclusive value='1'/>
    <xs:maxInclusive value='4'/>
  </xs:restriction>
</xs:simpleType>
```

6.14 SecurityToken

```
<xs:complexType name="SecurityToken">
  <xs:sequence>
    <xs:any minOccurs="1" maxOccurs="1" namespace="##any" processContents="lax"/>
  </xs:sequence>
</xs:complexType>
```

6.15 SupportedOperations

```
<xs:complexType name="SupportedOperations">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="IsXMLFilteringEnabled" type="xs:boolean" />
    <xs:element minOccurs="1" maxOccurs="1" name="IsJSONFilteringEnabled" type="xs:boolean" />
    <xs:element minOccurs="0" maxOccurs="1" name="SupportedContentFilteringLanguages"
type="isbm:SupportedContentFilteringLanguages" />
    <xs:element minOccurs="1" maxOccurs="1" name="SupportedAuthentications">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="1" name="SoapSupportedTokenSchemas">
            <xs:complexType>
              <xs:sequence>
                <xs:element minOccurs="1" maxOccurs="unbounded" name="TokenSchema"
type="isbm:TokenSchema" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element minOccurs="0" maxOccurs="1" name="RestSupportedAuthenticationSchemes">
            <xs:complexType>
              <xs:sequence>
                <xs:element minOccurs="1" maxOccurs="unbounded" name="AuthenticationScheme"
type="isbm:AuthenticationScheme" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element minOccurs="1" maxOccurs="1" name="SecurityLevelConformance" type="isbm:SecurityLevel"
/>
    <xs:element minOccurs="1" maxOccurs="1" name="IsDeadLetteringEnabled" type="xs:boolean" />
    <xs:element minOccurs="1" maxOccurs="1" name="IsChannelCreationEnabled" type="xs:boolean" />
    <xs:element minOccurs="1" maxOccurs="1" name="IsOpenChannelSecuringEnabled" type="xs:boolean" />
    <xs:element minOccurs="1" maxOccurs="1" name="IsWhitelistRequired" type="xs:boolean" />
    <xs:element minOccurs="1" maxOccurs="1" name="DefaultExpiryDuration" type="xs:duration"
nillable="true" />
    <xs:element minOccurs="1" maxOccurs="1" name="AdditionalInformationURL" type="xs:anyURI" />
  </xs:sequence>
</xs:complexType>
```

6.16 TokenSchema

```
<xs:complexType name="TokenSchema">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="NamespaceName">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="SchemaLocation" type="xs:anyURI" use="optional" />
          </xs:extension>
```

```
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

7 JSON Data Structures

The following data structures are used by the services defined in Service Definitions and are defined using JSON Schema.

7.1 AuthenticationScheme

```
"AuthenticationScheme": {
  "type": "object",
  "properties": {
    "schemeName": {
      "type": "string"
    },
    "schemeInfoUrl": {
      "type": "string",
      "format": "uri"
    }
  },
  "required": [
    "schemeName"
  ]
}
```

7.2 Channel

```
"Channel": {
  "type": "object",
  "properties": {
    "uri": {
      "type": "string",
      "format": "uri"
    },
    "channelType": {
      "$ref": "#/ChannelType"
    },
    "description": {
      "type": "string"
    },
    "securityTokens": {
      "description": "This can be provided when creating a channel but must never be returned.",
      "type": "array",
      "items": {
        "$ref": "#/SecurityToken"
      }
    }
  },
  "required": [
    "uri",
    "channelType"
  ]
}
```

7.3 ChannelType

```
"ChannelType": {
  "type": "string",
  "enum": [
    "Publication",
    "Request"
  ]
}
```

7.4 ContentFilteringLanguage

```
"ContentFilteringLanguage": {
  "type": "object",
  "properties": {
    "applicableMediaTypes": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "languageName": {
      "type": "string"
    },
    "languageVersion": {
      "type": "string"
    }
  },
  "required": [
    "applicableMediaTypes",
    "languageName"
  ]
}
```

7.5 Faults

All faults have the same basic schema:

```
{
  "type": "object",
  "properties": {
    "fault": {
      "type": "string"
    }
  },
  "required": [
    "fault"
  ]
}
```

7.6 FilterExpression

```
"FilterExpression": {
  "type": "object",
  "properties": {
    "applicableMediaTypes": {
      "type": "array",
      "items": {
        "type": "string"
      }
```

```
    },
    "expressionString": {
      "type": "object",
      "properties": {
        "expression": {
          "type": "string"
        },
        "language": {
          "type": "string"
        },
        "languageVersion": {
          "type": "string"
        }
      },
      "required": [
        "expression",
        "language"
      ]
    },
    "namespaces": {
      "type": "array",
      "items": {
        "$ref": "#/Namespace"
      }
    }
  },
  "required": [
    "expressionString"
  ]
}
```

7.7 Message

```
"Message": {
  "type": "object",
  "description": "messageId is returned on success; messageType is implicit based on the context.",
  "properties": {
    "messageId": {
      "type": "string"
    },
    "messageType": {
      "$ref": "#/MessageType"
    },
    "messageContent": {
      "$ref": "#/MessageContent"
    },
    "topics": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "minItems": 1
    },
    "expiry": {
      "type": "string",
      "format": "duration",
      "description": "Duration as defined by XML Schema xs:duration, http://w3c.org/TR/xmlschema-
2/#duration",
      "pattern": "[-]?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+([.][0-9]+)?S)?)?"
    },
```

```
    "requestMessageId": {
      "type": "string",
      "description": "Only valid for Response messages."
    },
    "originalMessageId": {
      "type": "string",
      "description": "Only valid if a message is being forwarded to another channel (when posting),
or the message is such message (on reading). Must be preserved on further forwarding of the message."
    }
  }
}
```

7.8 MessageContent

```
"MessageContent": {
  "type": "object",
  "properties": {
    "mediaType": {
      "type": "string"
    },
    "contentEncoding": {
      "type": "string"
    },
    "content": {
      "oneOf": [
        {
          "type": "string"
        },
        {
          "type": "object",
          "additionalProperties": true
        }
      ]
    }
  },
  "required": [
    "content"
  ]
}
```

7.9 MessageType

```
"MessageType": {
  "type": "string",
  "enum": [
    "Request",
    "Response",
    "Publication"
  ]
}
```

7.10 Namespace

```
"Namespace": {
  "type": "object",
  "properties": {
    "prefix": {
      "type": "string"
    },
```

```
    "name": {
      "type": "string"
    }
  },
  "required": [
    "prefix",
    "name"
  ]
}
```

7.11 Notification

```
"Notification": {
  "type": "object",
  "description": "SessionID and MessageID are provided via the request URL.\nTopic and
RequestMessageID are mutually exclusive. Topic MUST NOT be used for consumer request session response
notification. RequestMessageID MUST only be used for consumer request session response
notification.",
  "properties": {
    "sessionId": {
      "type": "string"
    },
    "messageId": {
      "type": "string"
    },
    "topics": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "requestMessageId": {
      "type": "string"
    }
  }
}
```

7.12 SecurityToken

```
"SecurityToken": {
  "type": "object",
  "description": "Exact security token types are implementation specific. Support must be provided
for at least UsernameToken.",
  "additionalProperties": true
}
```

7.13 SecurityDetails

```
"SecurityDetails": {
  "type": "object",
  "properties": {
    "isTLSEnabled": {
      "type": "boolean"
    },
    "isSecurityTokenRequired": {
      "type": "boolean"
    },
    "isSecurityTokenEncryptionEnabled": {
      "type": "boolean"
    },
```

```
    "isCertificateRequired": {
      "type": "boolean"
    },
    "isRBACEnabled": {
      "type": "boolean"
    },
    "isKeyManagementServiceEnabled": {
      "type": "boolean"
    },
    "isEndToEndMessageEncryptionEnabled": {
      "type": "boolean"
    }
  },
  "required": [
    "isTLSEnabled",
    "isSecurityTokenRequired",
    "isSecurityTokenEncryptionEnabled",
    "isCertificateRequired",
    "isRBACEnabled",
    "isKeyManagementServiceEnabled",
    "isEndToEndMessageEncryptionEnabled"
  ]
}
```

7.14 SecurityLevel

```
"SecurityLevel": {
  "type": "number",
  "enum": [ 1, 2, 3, 4 ]
}
```

7.15 Session

```
"Session": {
  "type": "object",
  "description": "sessionType is implicit based on the context.",
  "properties": {
    "sessionId": {
      "type": "string"
    },
    "sessionType": {
      "$ref": "#/SessionType"
    },
    "listenerUrl": {
      "type": "string",
      "format": "uri"
    },
    "topics": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "minItems": 1
    },
    "filterExpressions": {
      "type": "array",
      "items": {
        "$ref": "#/FilterExpression"
      }
```

```
    },
    "expirationListenerUrl": {
      "type": "string",
      "format": "uri"
    }
  }
}
```

7.16 SessionType

```
"SessionType": {
  "type": "string",
  "enum": [
    "PublicationProvider",
    "PublicationConsumer",
    "RequestProvider",
    "RequestConsumer"
  ]
}
```

7.17 SupportedOperations

```
"SupportedOperations": {
  "type": "object",
  "properties": {
    "isXMLFilteringEnabled": {
      "type": "boolean"
    },
    "isJSONFilteringEnabled": {
      "type": "boolean"
    },
    "supportedContentFilteringLanguages": {
      "type": "object",
      "properties": {
        "contentFilteringLanguages": {
          "type": "array",
          "items": {
            "$ref": "#/ContentFilteringLanguage"
          }
        }
      },
      "required": [
        "contentFilteringLanguages"
      ]
    },
    "supportedAuthentications": {
      "type": "object",
      "properties": {
        "soapSupportedTokenSchemas": {
          "type": "array",
          "items": {
            "$ref": "#/TokenSchema"
          }
        },
        "restSupportedAuthenticationSchemes": {
          "type": "array",
          "items": {
            "$ref": "#/AuthenticationScheme"
          }
        }
```

```
      }
    },
    "securityLevelConformance": {
      "$ref": "#/SecurityLevel"
    },
    "isDeadLetteringEnabled": {
      "type": "boolean"
    },
    "isChannelCreationEnabled": {
      "type": "boolean"
    },
    "isOpenChannelSecuringEnabled": {
      "type": "boolean"
    },
    "isWhitelistRequired": {
      "type": "boolean"
    },
    "defaultExpiryDuration": {
      "description": "Duration as defined by XML Schema xs:duration, http://w3c.org/TR/xmlschema-
2/#duration, or null",
      "oneOf": [
        {
          "type": "null"
        },
        {
          "type": "string",
          "format": "duration",
          "pattern": "[-]?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+([.][0-
9]+)?S)?)?"
        }
      ]
    },
    "additionalInformationURL": {
      "type": "string",
      "format": "uri"
    }
  },
  "required": [
    "isXMLFilteringEnabled",
    "isJSONFilteringEnabled",
    "supportedContentFilteringLanguages",
    "supportedAuthentications",
    "securityLevelConformance",
    "isDeadLetteringEnabled",
    "isChannelCreationEnabled",
    "isOpenChannelSecuringEnabled",
    "isWhitelistRequired",
    "defaultExpiryDuration",
    "additionalInformationURL"
  ]
}
```

7.18 TokenSchema

```
"TokenSchema": {
  "type": "object",
  "properties": {
    "namespaceName": {
      "type": "string"
    },
    "schemaLocation": {
```

```
      "type": "string",
      "format": "uri"
    }
  },
  "required": [
    "namespaceName"
  ]
}
```

7.19 UsernameToken

```
"UsernameToken": {
  "type": "object",
  "allOf": [
    {
      "$ref": "#/SecurityToken"
    }
  ],
  "properties": {
    "username": {
      "type": "string"
    },
    "password": {
      "type": "string",
      "format": "password"
    }
  },
  "required": [
    "password",
    "username"
  ]
}
```

7.20 Expiration

```
"Expiration": {
  "type": "object",
  "description": "SessionID and MessageID are provided via the request URL.\nOriginalMessageID is to
be provided if the message being expired was forwarded between channels.",
  "properties": {
    "sessionId": {
      "type": "string"
    },
    "messageId": {
      "type": "string"
    },
    "originalMessageId": {
      "type": "string"
    }
  }
}
```

8 Security Architecture

The general Service Requirements only provide security requirements related to authenticating operations against the channels on which they will be performed. This section considers security from an inter-enterprise context. It defines 4 levels of security to which ISBM implementations may conform. In this version of the specification, these security levels are introduced and briefly discussed. In future revisions of this specification, the following security

levels will be associated with concrete requirements for the services in general and requirements for specific services where necessary.

8.1 Security Level 1 – None

Security Level 1 is characterized by fulfilling no security criteria. That is:

- SSL/TLS are NOT used for transport layer security
- Security tokens are NOT used to secure channels, or tokens are exchanged in the clear without encryption
- Security tokens MAY or MAY NOT be stored encrypted, if used
- Certificates are NOT used for confirming identity

8.1.1 Usage Scenarios

This security level is NOT RECOMMENDED for production environments. However, it MAY be suitable for use in development and testing environments. It MAY also be used in known restricted environments, such as isolated networks.

8.2 Security Level 2 – Core Security

Security Level 2, Core Security, provides a basic set of security requirements. In contrast to Security level 1, this security level is characterized by providing transport layer security and securing tokens at rest, that is:

- All the communications MUST use transport layer security, e.g., SSL/TLS
- Security tokens MAY be used but MUST be are stored encrypted by the ISBM Service Provider
- Best practices are used to exchange/configure security tokens out-of-band

The Core Security level MAY also utilize Role-Based Access Control for configuring the services and performing their operations.

8.2.1 Usage Scenarios

The Core Security level MAY be used for production environments and is most appropriate for *intra*-enterprise connectivity. It is NOT RECOMMENDED for environments, even intra-enterprise environments, that require a higher level of security.

8.3 Security Level 3 – Inter-Enterprise Security

Security Level 3, or Inter-Enterprise Security, includes all the requirements of Core Security and adds requirements suitable for an inter-enterprise context. In particular, this level of security is characterized by confirming the identity of interacting systems. The additional requirements to the meet the Inter-Enterprise Security level include:

- Certificates (or other mechanisms) MUST be used to identify ISBM Service Providers and Clients, including Notification endpoints specified by subscription sessions.
- Role-Based Access Control MUST be used for configuration of the Service Provider and performing the operations on the Services.
- All channels MUST be configured with security tokens and non-authenticated operations (GetChannels and CreateChannel) only respond to recognized systems based on the identity check (such as checking their certificates).

At this level of security, ISBM Service Providers MAY use 3rd party services (Key Management Services) to encrypt/decrypt security tokens on demand.

8.3.1 Usage Scenarios

The Inter-Enterprise Security Level is RECOMMENDED for most Inter-Enterprise scenarios. It is NOT RECOMMENDED for use in highly secure environments that require additional security guarantees.

8.4 Security Level 4 – Defense

The Defense Security Level, Security Level 4, includes all the requirements of Inter-Enterprise Security (and Core Security) with the addition of requirements necessary for highly secure environments, such as National Defense Services. This level of security is characterized by securing the messages and other data within the ISBM Service Provider. Additional requirements for the Defense Security Level include:

- Full end-to-end encryption of messages MUST be performed, that is, the message content is encrypted on the server/s of the ISBM Service Provider
- Security keys (for messages encryption) and security tokens (for channel access) MUST be stored encrypted
- It is RECOMMENDED that 3rd party KMSs (Key Management Services) be used to encrypt/decrypt the security keys and tokens.
- Access to an ISBM Service Provider at this level of security by systems of lower-security levels MUST be performed using appropriate negotiation protocols.

NOTE A typical approach to encrypting a message may be as follows: a random encryption key is created for each message; the message is encrypted using its key and the encrypted message is stored; the key is encrypted using a 3rd party KMS (to prevent the 3rd party from the seeing the confidential messages); the encrypted key is then stored in the database.

8.4.1 Usage Scenarios

The Defense Security Level is RECOMMENDED for highly secure environments such as those often required by Defense. This may be both intra- and inter-enterprise scenarios depending on the requirements of the deployment.

8.5 Security Level Matrix

The following table summarizes the four levels of security discussed above.

	Transport Layer Security	Uses Security Token	Encrypts Security Tokens	Uses Identity Certificate	Role-Based Access Control	3rd Party KMS	End-to-end encryption?	Suitable for
Security Level 1	False	True/False	False	False	False	False	False	Development and Testing environment
Security Level 2	True	True	True	False	True/False	False	False	Intra-enterprise connectivity
Security Level 3	True	True	True	True	True	True/False	False	Inter-Enterprise connectivity
Security Level 4	True	True	True	True	True	True	True	Highly secure environments

9 Conformance

Any assessment of conformance of an ISBM implementation MUST be qualified by the following:

1. Support for the Channel Management Service
2. Support for the Notification Service
3. Support for the Expiration Listener Service
4. Support for the Provider Publication Service
5. Support for the Consumer Publication Service
6. Support for the Provider Request Service
7. Support for the Consumer Request Service
8. Support for Message Forwarding and Traceability (OriginalMessageID field)
9. Support for SOAP 1.1 and SOAP 1.2 services
10. Support for HTTP 1.1
11. Support for OpenAPI 3.0.1 services
12. Support for Filter Expressions in an XPath 1.0 format for XML content
13. Support for Filter Expressions in an JSONPath format for JSON content
14. Support for transport layer security (e.g. SSL/TLS) in order to secure tokens and messages, and to prevent replay attacks.
15. Support for Security Tokens using WS-Security UsernameToken
16. Support for HTTP basic and/or digest authentication and authorization
17. Support for other Security Tokens formats (including HTTP authentication/authorization token formats)
18. A statement of the total conformance concerning services and security methods supported or, in case of partial conformance, a statement identifying explicitly the areas of non-conformance

Annex A. Specification Files

The following lists the files containing the Web Services descriptions for SOAP (WSDL format) and REST (OpenAPI format).

A.1 OpenAPI Definitions

http://www.openoandm.org/isbm/2.1/openapi/channel_management_service.yml

http://www.openoandm.org/isbm/2.1/openapi/channel_management_service.json

http://www.openoandm.org/isbm/2.1/openapi/notification_service.yml

http://www.openoandm.org/isbm/2.1/openapi/notification_service.json

http://www.openoandm.org/isbm/2.1/openapi/provider_publication_service.yml

http://www.openoandm.org/isbm/2.1/openapi/provider_publication_service.json

http://www.openoandm.org/isbm/2.1/openapi/consumer_publication_service.yml

http://www.openoandm.org/isbm/2.1/openapi/consumer_publication_service.json

http://www.openoandm.org/isbm/2.1/openapi/provider_request_service.yml

http://www.openoandm.org/isbm/2.1/openapi/provider_request_service.json

http://www.openoandm.org/isbm/2.1/openapi/consumer_request_service.yml

http://www.openoandm.org/isbm/2.1/openapi/consumer_request_service.json

http://www.openoandm.org/isbm/2.1/openapi/configuration_discovery_service.yml

http://www.openoandm.org/isbm/2.1/openapi/configuration_discovery_service.json

http://www.openoandm.org/isbm/2.1/openapi/isbm_complete.yml

http://www.openoandm.org/isbm/2.1/openapi/isbm_complete.json

A.2 WSDLs

http://www.openoandm.org/isbm/2.1/wsdl/ChannelManagementService.wsdl

http://www.openoandm.org/isbm/2.1/wsdl/NotificationService.wsdl

http://www.openoandm.org/isbm/2.1/wsdl/ProviderPublicationService.wsdl

http://www.openoandm.org/isbm/2.1/wsdl/ConsumerPublicationService.wsdl

http://www.openoandm.org/isbm/2.1/wsdl/ProviderRequestService.wsdl

http://www.openoandm.org/isbm/2.1/wsdl/ConsumerRequestService.wsdl

http://www.openoandm.org/isbm/2.1/wsdl/ConfigurationDiscoveryService.wsdl

NOTE Where the schema of an inidivdiual service definition is unchanged from prior versions, the earliest prior version to which it is identical is used as the version specifier. For example, the schema of the Channel Management Service may indicate version 2.0 as it is unchanged in the 2.1 specification.

A.3 Packaged Specification

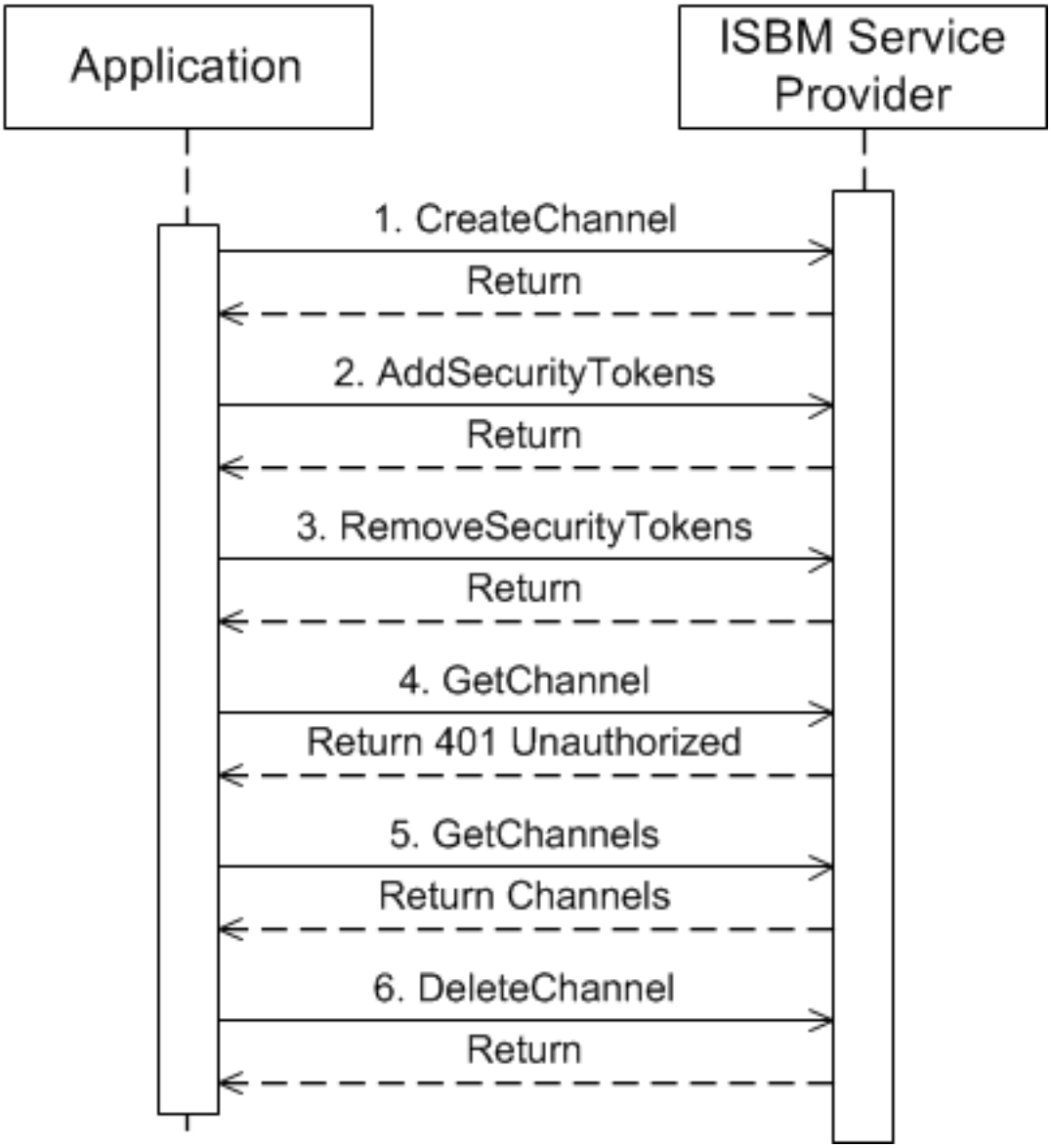
http://www.openoandm.org/isbm/isbm-soap-2.1.zip

http://www.openoandm.org/isbm/isbm-rest-2.1.zip

http://www.openoandm.org/isbm/isbm-all-2.1.zip

Annex B. Example HTTP Flows

B.1 Channel Management Example



B.1.1 CreateChannel

The Application creates a channel on the ISBM Service Provider and assigns a WS-Security security token.

NOTE XML special characters must be escaped, as seen with the **<** character in the `Password` element.

B.1.1.1 HTTP Request

```
POST /ChannelManagementService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 705
SOAPAction: "http://www.openoandm.org/isbm/CreateChannel"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:CreateChannel xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
      <isbm:ChannelType>Publication</isbm:ChannelType>
      <isbm:SecurityToken>
        <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
          <wsse:Username>Application1</wsse:Username>
          <wsse:Password>&lt;s9.vQfLDx9LgL</wsse:Password>
        </wsse:UsernameToken>
      </isbm:SecurityToken>
    </isbm:CreateChannel>
  </soap:Body>
</soap:Envelope>
```

B.1.1.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 238

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:CreateChannelResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.1.2 AddSecurityToken

The Application assigns an additional security token to the channel.

B.1.2.1 HTTP Request

```
POST /ChannelManagementService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 892
SOAPAction: "http://www.openoandm.org/isbm/AddSecurityToken"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Application1</wsse:Username>
        <wsse:Password>&lt;s9.vQfLDx9LgL</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:AddSecurityToken xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
      <isbm:SecurityToken>
        <wsse:UsernameToken>
          <wsse:Username>Application2</wsse:Username>
          <wsse:Password>chHM?rFum{48mg</wsse:Password>
```

```
        </wsse:UsernameToken>
      </isbm:SecurityToken>
    </isbm:AddSecurityToken>
  </soap:Body>
</soap:Envelope>
```

B.1.2.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 241

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:AddSecurityTokenResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.1.3 RemoveSecurityToken

The Application removes the original security token from the channel.

B.1.3.1 HTTP Request

```
POST /ChannelManagementService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 898
SOAPAction: "http://www.openoandm.org/isbm/RemoveSecurityToken"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Application2</wsse:Username>
        <wsse:Password>chHM?rFum{48mg</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:RemoveSecurityToken xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
      <isbm:SecurityToken>
        <wsse:UsernameToken>
          <wsse:Username>Application1</wsse:Username>
          <wsse:Password>&lt;s9.vQfLDx9LgL</wsse:Password>
        </wsse:UsernameToken>
      </isbm:SecurityToken>
    </isbm:RemoveSecurityToken>
  </soap:Body>
</soap:Envelope>
```

B.1.3.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 244
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:RemoveSecurityTokenResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.1.4 GetChannel

The Application attempts to retrieve channel information using the original security token and receives an authorization failure.

B.1.4.1 HTTP Request

```
POST /ChannelManagementService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 656
SOAPAction: "http://www.openoandm.org/isbm/GetChannel"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>Application1</wsse:Username>
        <wsse:Password>&lt;s9.vQfLDx9LgL</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:GetChannel xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
    </isbm:GetChannel>
  </soap:Body>
</soap:Envelope>
```

B.1.4.2 HTTP Response

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset=utf-8
Content-Length: 401

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>Channel is not accessible.</faultstring>
      <detail>
        <isbm:ChannelFault xmlns:isbm="http://www.openoandm.org/isbm/"/>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

B.1.5 GetChannels

The Application retrieves information about channels filtered by the newly assigned security token.

B.1.5.1 HTTP Request

```
POST /ChannelManagementService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 559
SOAPAction: "http://www.openoandm.org/isbm/GetChannels"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>Application2</wsse:Username>
        <wsse:Password>chHM?rFum{48mg</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:GetChannels xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.1.5.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 442

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:GetChannelsResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:Channel>
        <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
        <isbm:ChannelType>Publication</isbm:ChannelType>
      </isbm:Channel>
    </isbm:GetChannelsResponse>
  </soap:Body>
</soap:Envelope>
```

B.1.6 DeleteChannel

The Application removes the channel from the isbm Service Provider.

B.1.6.1 HTTP Request

```
POST /ChannelManagementService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 659
SOAPAction: "http://www.openoandm.org/isbm/DeleteChannel"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>Application2</wsse:Username>
```

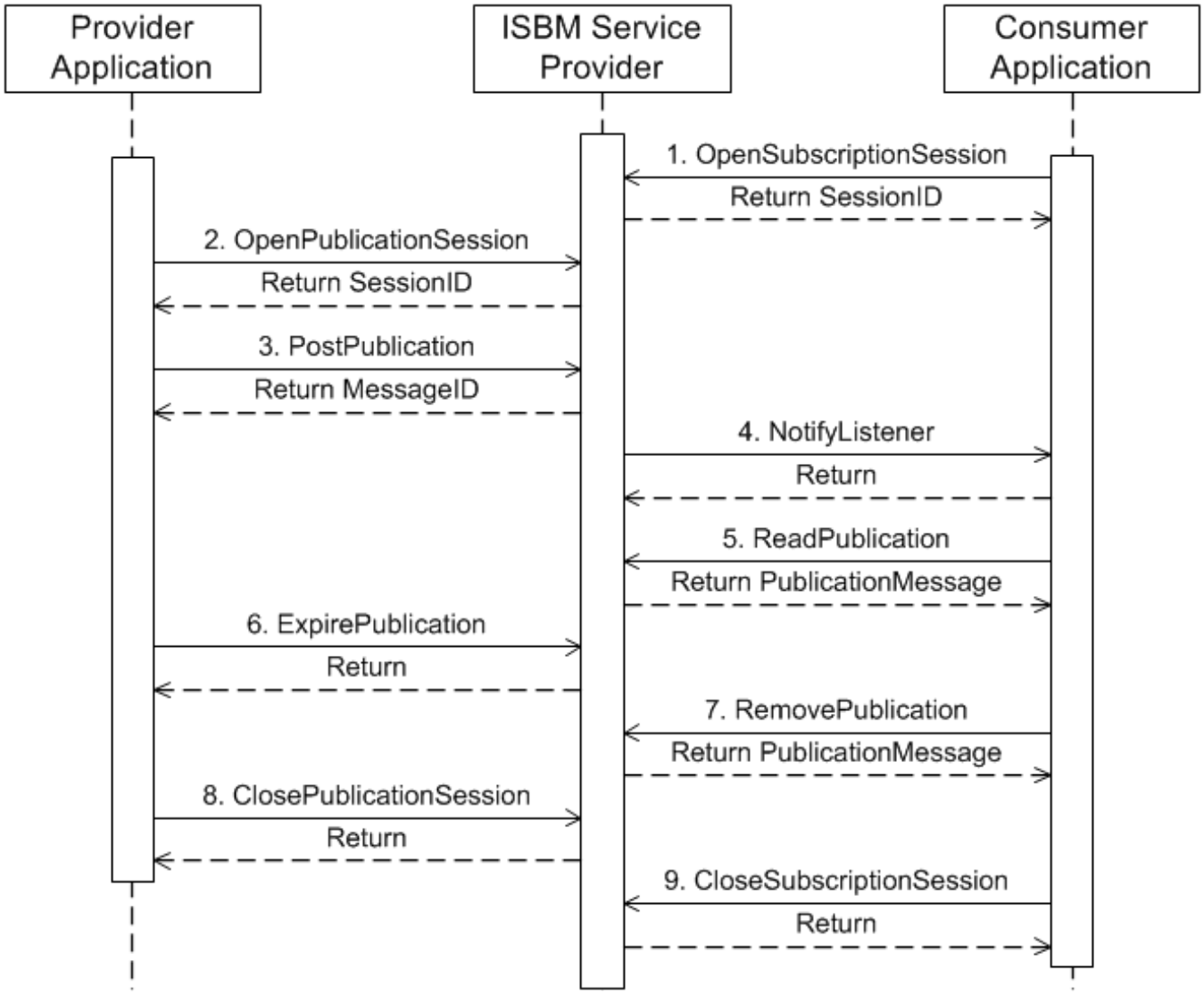
```
        <wsse:Password>chHM?rFum{48mg</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:DeleteChannel xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
    </isbm:DeleteChannel>
  </soap:Body>
</soap:Envelope>
```

B.1.6.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 238

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:DeleteChannelResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.2 Publish-Subscribe Example



B.2.1 OpenSubscriptionSession

The Consumer Application opens a subscription session with the ISBM Service Provider and receives a session identifier.

B.2.1.1 HTTP Request

```
POST /ConsumerPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 812
SOAPAction: "http://www.openoandm.org/isbm/OpenSubscriptionSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
```

```
      <wsse:UsernameToken>
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>Dj8(bCU)4bnhjc</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:OpenSubscriptionSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
      <isbm:Topic>Text</isbm:Topic>
      <isbm:ListenerURL>http://consumer.example.com/NotificationService</isbm:ListenerURL>
    </isbm:OpenPublicationSession>
  </soap:Body>
</soap:Envelope>
```

B.2.1.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 366

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:OpenSubscriptionSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>e94c645a-6450-411e-8ec7-4b70620d3a98</isbm:SessionID>
    </isbm:OpenSubscriptionSessionResponse>
  </soap:Body>
</soap:Envelope>
```

B.2.2 OpenPublicationSession

The Provider Application opens a publication session with the ISBM Service Provider and receives a session identifier.

B.2.2.1 HTTP Request

```
POST /ProviderPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 684
SOAPAction: "http://www.openoandm.org/isbm/OpenPublicationSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>qEJaz4F?U4rW;q</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:OpenPublicationSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
    </isbm:OpenPublicationSession>
  </soap:Body>
</soap:Envelope>
```

B.2.2.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 364

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:OpenPublicationSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>ac0ee730-ca88-421a-b348-ce0a1babdb1c</isbm:SessionID>
    </isbm:OpenPublicationSessionResponse>
  </soap:Body>
</soap:Envelope>
```

B.2.3 PostPublication

The Provider Application posts a publication message to the ISBM Service Provider and receives a message identifier.

B.2.3.1 HTTP Request

```
POST /ProviderPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 799
SOAPAction: "http://www.openoandm.org/isbm/PostPublication"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>qEJaz4F?U4rW;q</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:PostPublication xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>ac0ee730-ca88-421a-b348-ce0a1babdb1c</isbm:SessionID>
      <isbm:MessageContent xsi:type="isbm:XMLContent">
        <Content>Hello World!</Content>
      </isbm:MessageContent>
      <isbm:Topic>Text</isbm:Topic>
    </isbm:PostPublication>
  </soap:Body>
</soap:Envelope>
```

B.2.3.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 350

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:PostPublicationResponse xmlns:isbm="http://www.openoandm.org/isbm/">
```

```
      <isbm:MessageID>8007a3fa-70e3-4e90-a2b9-d8469cae2e5a</isbm:MessageID>
    </isbm:PostPublicationResponse>
  </soap:Body>
</soap:Envelope>
```

B.2.4 NotifyListener

The ISBM Service Provider notifies the Consumer Application of an applicable publication message.

B.2.4.1 HTTP Request

```
POST /NotifyListener HTTP/1.1
Host: consumer.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 444
SOAPAction: "http://www.openoandm.org/isbm/NotifyListener"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:NotifyListener xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>e94c645a-6450-411e-8ec7-4b70620d3a98</isbm:SessionID>
      <isbm:MessageID>8007a3fa-70e3-4e90-a2b9-d8469cae2e5a</isbm:MessageID>
      <isbm:Topic>Text</isbm:Topic>
    </isbm:NotifyListener>
  </soap:Body>
</soap:Envelope>
```

B.2.4.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 239

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:NotifyListenerResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.2.5 ReadPublication

The Consumer Application reads the publication message from the ISBM Service Provider.

B.2.5.1 HTTP Request

```
POST /ConsumerPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 672
SOAPAction: "http://www.openoandm.org/isbm/ReadPublication"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ConsumerApplication</wsse:Username>
```

```
        <wsse:Password>Dj8(bCU)4bnhjc</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:ReadPublication xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>e94c645a-6450-411e-8ec7-4b70620d3a98</isbm:SessionID>
    </isbm:ReadPublication>
  </soap:Body>
</soap:Envelope>
```

B.2.5.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 552

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:ReadPublicationResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:PublicationMessage>
        <isbm:MessageID>8007a3fa-70e3-4e90-a2b9-d8469cae2e5a</isbm:MessageID>
      <isbm:MessageContent xsi:type="isbm:XMLContent">
        <Content>Hello World!</Content>
      </isbm:MessageContent>
        <isbm:Topic>Text</isbm:Topic>
      </isbm:PublicationMessage>
    </isbm:ReadPublicationResponse>
  </soap:Body>
</soap:Envelope>
```

B.2.6 ExpirePublication

The Provider Application manually expires the publication message from the ISBM Service Provider. The message is still visible to the Consumer Application since it has already been read.

B.2.6.1 HTTP Request

```
POST /ProviderPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 752
SOAPAction: "http://www.openoandm.org/isbm/ExpirePublication"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>qEJaz4F?U4rW;q</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:ExpirePublication xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>ac0ee730-ca88-421a-b348-ce0a1babdb1c</isbm:SessionID>
      <isbm:MessageID>8007a3fa-70e3-4e90-a2b9-d8469cae2e5a</isbm:MessageID>
    </isbm:ExpirePublication>
```

```
  </soap:Body>
</soap:Envelope>
```

B.2.6.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 242

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:ExpirePublicationResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.2.7 RemovePublication

The Consumer Application removes the publication message from the ISBM Service Provider.

B.2.7.1 HTTP Request

```
POST /ConsumerPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 677
SOAPAction: "http://www.openoandm.org/isbm/RemovePublication"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>Dj8(bCU)4bnhjc</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:RemovePublication xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>e94c645a-6450-411e-8ec7-4b70620d3a98<</isbm:SessionID>
    </isbm:RemovePublication>
  </soap:Body>
</soap:Envelope>
```

B.2.7.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 242

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:RemovePublicationResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.2.8 ClosePublicationSession

The Provider Application closes the publication session with the ISBM Service Provider.

B.2.8.1 HTTP Request

```
POST /ProviderPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 688
SOAPAction: "http://www.openoandm.org/isbm/ClosePublicationSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>qEJaz4F?U4rW;q</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:ClosePublicationSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>ac0ee730-ca88-421a-b348-ce0a1babdb1c</isbm:SessionID>
    </isbm:ClosePublicationSession>
  </soap:Body>
</soap:Envelope>
```

B.2.8.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 248

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:ClosePublicationSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.2.9 CloseSubscriptionSession

The Consumer Application closes the subscription session with the ISBM Service Provider.

B.2.9.1 HTTP Request

```
POST /ConsumerPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 690
SOAPAction: "http://www.openoandm.org/isbm/ClosePublicationSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
```

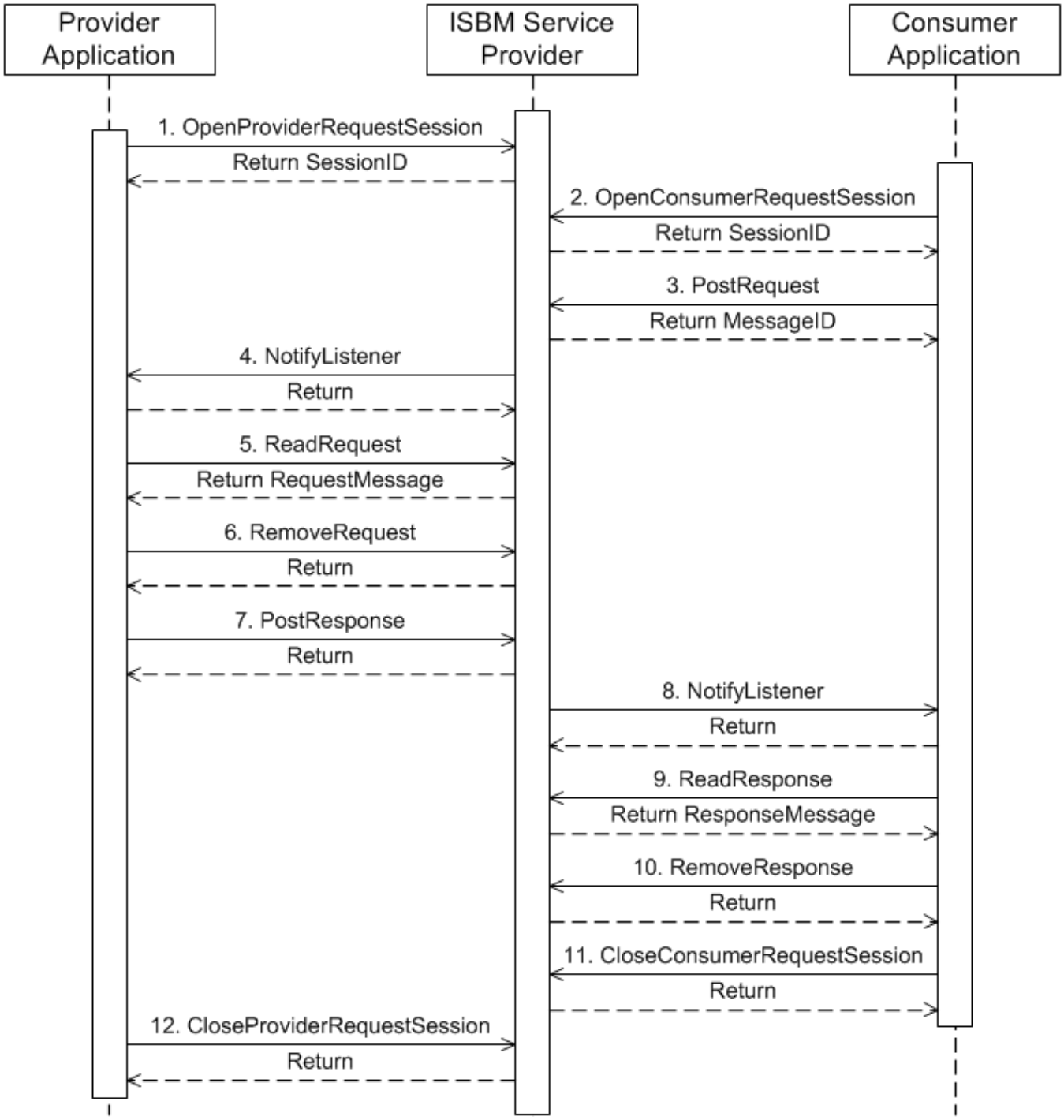
```
      <wsse:UsernameToken>
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>Dj8(bCU)4bnhjc</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:CloseSubscriptionSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>e94c645a-6450-411e-8ec7-4b70620d3a98</isbm:SessionID>
    </isbm:CloseSubscriptionSession>
  </soap:Body>
</soap:Envelope>
```

B.2.9.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 249

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:CloseSubscriptionSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.3 Request-Response Example



B.3.1 OpenProviderRequestSession

The Provider Application opens a provider request session with the ISBM Service Provider and receives a session identifier.

B.3.1.1 HTTP Request

```
POST /ProviderRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 819
SOAPAction: "http://www.openoandm.org/isbm/OpenProviderRequestSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>9gy#gXENxph8?W</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:OpenProviderRequestSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
      <isbm:Topic>Text</isbm:Topic>
      <isbm:ListenerURL>http://provider.example.com/NotificationService</isbm:ListenerURL>
    </isbm:OpenProviderRequestSession>
  </soap:Body>
</soap:Envelope>
```

B.3.1.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 372

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:OpenProviderRequestSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>16ee00f6-8fa9-4e80-8c36-a9d6d2bdb551</isbm:SessionID>
    </isbm:OpenProviderRequestSessionResponse>
  </soap:Body>
</soap:Envelope>
```

B.3.2 OpenConsumerRequestSession

The Consumer Application opens a consumer request session with the ISBM Service Provider and receives a session identifier.

B.3.2.1 HTTP Request

```
POST /ConsumerRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 783
SOAPAction: "http://www.openoandm.org/isbm/OpenConsumerRequestSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
```

```
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>^Um.7oFM9jrnnC</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:OpenConsumerRequestSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ChannelURI>/Enterprise/Site/Area/WorkCenter</isbm:ChannelURI>
      <isbm:ListenerURL>http://consumer.example.com/NotificationService</isbm:ListenerURL>
    </isbm:OpenConsumerRequestSession>
  </soap:Body>
</soap:Envelope>
```

B.3.2.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 372

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:OpenConsumerRequestSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>a9b5c71a-d1b5-4fc8-81d1-ba1fee3af0df</isbm:SessionID>
    </isbm:OpenConsumerRequestSessionResponse>
  </soap:Body>
</soap:Envelope>
```

B.3.3 PostRequest

The Consumer Application posts a request message to the ISBM Service Provider and receives a message identifier.

B.3.3.1 HTTP Request

```
POST /ConsumerRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 748
SOAPAction: "http://www.openoandm.org/isbm/PostRequest"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>^Um.7oFM9jrnnC</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:PostRequest xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>a9b5c71a-d1b5-4fc8-81d1-ba1fee3af0df</isbm:SessionID>
      <isbm:MessageContent xsi:type="StringContent" mediaType="text/xml">
        <Content>Ping!</Content>
      </isbm:MessageContent>
    </isbm:PostRequest>
```

```
  </soap:Body>
</soap:Envelope>
```

B.3.3.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 342

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:PostRequestResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:MessageID>e8cfecb1-d2fc-4167-88f7-c90d60fc53ee</isbm:MessageID>
    </isbm:PostRequestResponse>
  </soap:Body>
</soap:Envelope>
```

B.3.4 NotifyListener

The ISBM Service Provider notifies the Provider Application of an applicable request message.

B.3.4.1 HTTP Request

```
POST /NotifyListener HTTP/1.1
Host: provider.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 444
SOAPAction: "http://www.openoandm.org/isbm/NotifyListener"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:NotifyListener xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>16ee00f6-8fa9-4e80-8c36-a9d6d2bdb551</isbm:SessionID>
      <isbm:MessageID>e8cfecb1-d2fc-4167-88f7-c90d60fc53ee</isbm:MessageID>
      <isbm:Topic>Text</isbm:Topic>
    </isbm:NotifyListener>
  </soap:Body>
</soap:Envelope>
```

B.3.4.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 239

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:NotifyListenerResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.3.5 ReadRequest

The Provider Application reads the request message from the ISBM Service Provider.

B.3.5.1 HTTP Request

```
POST /ProviderRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 664
SOAPAction: "http://www.openoandm.org/isbm/ReadRequest"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>9gy#gXENxph8?W</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:ReadRequest xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>16ee00f6-8fa9-4e80-8c36-a9d6d2bdb551</isbm:SessionID>
    </isbm:ReadRequest>
  </soap:Body>
</soap:Envelope>
```

B.3.5.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 529

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:ReadRequestResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:RequestMessage>
        <isbm:MessageID>e8cfecb1-d2fc-4167-88f7-c90d60fc53ee</isbm:MessageID>
      <isbm:MessageContent xsi:type="StringContent" mediaType="text/xml">
        <Content>Ping!</Content>
      </isbm:MessageContent>
        <isbm:Topic>Text</isbm:Topic>
      </isbm:RequestMessage>
    </isbm:ReadRequestResponse>
  </soap:Body>
</soap:Envelope>
```

B.3.6 RemoveRequest

The Provider Application removes the request message from the ISBM Service Provider.

B.3.6.1 HTTP Request

```
POST /ProviderRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 669
SOAPAction: "http://www.openoandm.org/isbm/RemoveRequest"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
```

```
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>9gy#gXENxph8?W</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:RemoveRequest xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>16ee00f6-8fa9-4e80-8c36-a9d6d2bdb551<</isbm:SessionID>
    </isbm:RemoveRequest>
  </soap:Body>
</soap:Envelope>
```

B.3.6.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 238

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:RemoveRequestResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.3.7 PostResponse

The Provider Application posts a response message to the ISBM Service Provider.

B.3.7.1 HTTP Request

```
POST /ProviderRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 840
SOAPAction: "http://www.openoandm.org/isbm/PostResponse"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>9gy#gXENxph8?W</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:PostResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>16ee00f6-8fa9-4e80-8c36-a9d6d2bdb551</isbm:SessionID>
      <isbm:RequestMessageID>e8cfecb1-d2fc-4167-88f7-c90d60fc53ee</isbm:RequestMessageID>
      <isbm:MessageContent xsi:type="StringContent" mediaType="text/xml">
        <Content>Pong!</Content>
      </isbm:MessageContent>
    </isbm:PostResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

B.3.7.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 237

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:PostResponseResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.3.8 NotifyListener

The ISBM Service Provider notifies the Consumer Application of an applicable response message.

B.3.8.1 HTTP Request

```
POST /NotifyListener HTTP/1.1
Host: consumer.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 498
SOAPAction: "http://www.openoandm.org/isbm/NotifyListener"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:NotifyListener xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>a9b5c71a-d1b5-4fc8-81d1-ba1fee3af0df</isbm:SessionID>
      <isbm:MessageID>af250a33-d5af-4c25-bb57-56802d8fea79</isbm:MessageID>
      <isbm:RequestMessageID>e8cfecb1-d2fc-4167-88f7-c90d60fc53ee</isbm:RequestMessageID>
    </isbm:NotifyListener>
  </soap:Body>
</soap:Envelope>
```

B.3.8.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 239

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:NotifyListenerResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.3.9 ReadResponse

The Consumer Application reads the response message from the ISBM Service Provider.

B.3.9.1 HTTP Request

```
POST /ConsumerRequestService HTTP/1.1
Host: isbm.example.com
```

```
Content-Type: text/xml; charset=utf-8
Content-Length: 756
SOAPAction: "http://www.openoandm.org/isbm/ReadResponse"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>^Um.7oFM9jrnnC</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:ReadResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>a9b5c71a-d1b5-4fc8-81d1-ba1fee3af0df</isbm:SessionID>
      <isbm:RequestMessageID>e8cfecb1-d2fc-4167-88f7-c90d60fc53ee</isbm:RequestMessageID>
    </isbm:ReadResponse>
  </soap:Body>
</soap:Envelope>
```

B.3.9.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 495

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:ReadResponseResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:ResponseMessage>
        <isbm:MessageID>af250a33-d5af-4c25-bb57-56802d8fea79</isbm:MessageID>
      <isbm:MessageContent xsi:type="StringContent" mediaType="text/xml">
        <Content>Ping!</Content>
      </isbm:MessageContent>
      </isbm:ResponseMessage>
    </isbm:ReadResponseResponse>
  </soap:Body>
</soap:Envelope>
```

B.3.10 RemoveResponse

The Consumer Application removes the response message from the ISBM Service Provider.

B.3.10.1 HTTP Request

```
POST /ConsumerPublicationService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 760
SOAPAction: "http://www.openoandm.org/isbm/RemoveResponse"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
```

```
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>^Um.7oFM9jrnnC</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:RemoveResponse xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>a9b5c71a-d1b5-4fc8-81d1-ba1fee3af0df</isbm:SessionID>
      <isbm:RequestMessageID>e8cfecb1-d2fc-4167-88f7-c90d60fc53ee</isbm:RequestMessageID>
    </isbm:RemoveResponse>
  </soap:Body>
</soap:Envelope>
```

B.3.10.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 239

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:RemoveResponseResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.3.11 CloseConsumerRequestSession

The Consumer Application closes the consumer request session with the ISBM Service Provider.

B.3.11.1 HTTP Request

```
POST /ConsumerRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 696
SOAPAction: "http://www.openoandm.org/isbm/CloseConsumerRequestSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ConsumerApplication</wsse:Username>
        <wsse:Password>^Um.7oFM9jrnnC</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:CloseConsumerRequestSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>a9b5c71a-d1b5-4fc8-81d1-ba1fee3af0df</isbm:SessionID>
    </isbm:CloseConsumerRequestSession>
  </soap:Body>
</soap:Envelope>
```

B.3.11.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 252
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:CloseConsumerRequestSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

B.3.12 CloseProviderRequestSession

The Provider Application closes the provider request session with the ISBM Service Provider.

B.3.12.1 HTTP Request

```
POST /ProviderRequestService HTTP/1.1
Host: isbm.example.com
Content-Type: text/xml; charset=utf-8
Content-Length: 696
SOAPAction: "http://www.openoandm.org/isbm/CloseProviderRequestSession"

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>ProviderApplication</wsse:Username>
        <wsse:Password>9gy#gXENxph8?W</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <isbm:CloseProviderRequestSession xmlns:isbm="http://www.openoandm.org/isbm/">
      <isbm:SessionID>16ee00f6-8fa9-4e80-8c36-a9d6d2bdb551</isbm:SessionID>
    </isbm:CloseProviderRequestSession>
  </soap:Body>
</soap:Envelope>
```

B.3.12.2 HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 252

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <isbm:CloseProviderRequestSessionResponse xmlns:isbm="http://www.openoandm.org/isbm/"/>
  </soap:Body>
</soap:Envelope>
```

Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

- Alan T. Johnston, MIMOSA
- Chris Monchinski, ISA
- Georg Grossmann, University of South Australia
- James Fort, MIMOSA
- Pak Wong, PdMA Corporation
- Yan Lu, National Institute of Standards and Technology

Bibliography

[1] XML Path Language (XPath) Version 1.0, James Clark and Steven DeRose, Editors. World Wide Web Consortium, 16 Nov 1999. This version is *http://www.w3.org/TR/1999/REC-xpath-19991116*. The latest version is available at *http://www.w3.org/TR/xpath*.

[2] Web Services Description Language (WSDL) 1.1, E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Authors. World Wide Web Consortium, 15 March 2002. This version of the Web Services Description Language 1.1 Note is *http://www.w3.org/TR/2001/NOTE-wsdl-20010315*. The latest version of Web Services Description Language 1.1 is available at *http://www.w3.org/TR/wsdl*.

[3] Open API Initiative: OpenAPI Specification 3.0.2, *https://github.com/OAI/OpenAPISpecification/blob/master/versions/3.0.2.md*

[4] Representational State Transfer (REST), Fielding, Roy Thomas (2000). "Chapter 5: Representational State Transfer (REST)" available at *http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm*. *Architectural Styles and the Design of Network-based Software Architectures* (Ph.D.). University of California, Irvine.

[5] JSONPath, Stefan Goessner (2007), https://goessner.net/articles/JsonPath/