OpenO&M™
Information Service Bus Model (ISBM) Specification
v1.0 RC6

This document defines the OpenO&M Information Service Bus Model (ISBM). It defines a underlying logical data model, the web services for the registry, and a normative XML Schema/WSDL specification for the web services.

This specification defines an OpenO&M Information Service Bus Model (ISBM) for exchanging the information defined in the ISA 95 Enterprise/Control System Integration standards, OpenO&M Common Interoperability Registry (CIR), MIMOSA OSA-EAI, the WBF Business to Manufacturing Markup Language (B2MML), ISO 15926 information, and OPC UA address space objects that have been converted to standardized OPC UA XML payloads.

The ISBM defines a minimal interface subset to Enterprise Service Buses (ESB) and other XML message exchange middleware, using a standard interface consisting of channels and topics. The benefit from this approach is to allow applications to expose a single, standardized interface rather than having to be custom built for every version and format of ESB or message exchange system.

The knowledge requirements to interface to just one ESB can be immense, and is usually not transferable to a different ESB. The ISBM defines a single interface, independent of the underlying services, for Level 3-3 and Level 4-3 communications. This removes the need for vendors to build custom interface after custom interface, and for end users to get locked into a single vendor because their investment prevents them from reusing any of the integration efforts.

Table of Contents

1 References

- ANSI/ISA 95.00.01, Enterprise-Control System Integration – Part 1: Models and Terminology
- ANSI/ISA 95.00.02, Enterprise-Control System Integration – Part 2: Object Model Attributes
- ANSI/ISA 95.00.05, Enterprise-Control System Integration – Part 5: Business to Manufacturing Transactions
- IEC 62264-1, Enterprise-Control System Integration – Part 1: Models and Terminology
- IEC 62264-2, Enterprise-Control Systems Integration – Part 2: Object Model Attributes
- IEC 62264-5, Enterprise-Control Systems Integration – Part 5: Business to Manufacturing Transactions
- IEC 62541, OPC Unified Architecture, Parts 1-12, www.opcfoundation.org
- WBF B2MML Schemas, www.wbf.org, V0400 and later schemas
- MIMOSA Open Systems Architecture for Enterprise Application Integration (OSA-EAI) www.mimosa.org
- OpenO&M Common Interoperability Registry (CIR), www.mimosa.org
- ISO 15926, www.iso.org
- [X509] X.509 Public Key Certificate Infrastructure, http://www.itu.int/rec/T-REC-X.509-200003-I/e
- [IS Glossary] Internet Security Glossary, http://www.ietf.org/rfc/rfc2828.txt
- [NIST 800-12] Introduction to Computer Security, http://csrc.nist.gov/publications/nistpubs/800-12/
- ISA 14750, Information Technology – Open Distributed Processing – Interface Definition Language

2 OpenO&M Information Service Bus Model

2.1 Interface Model

The ISBM defines a standard set of services that would be provided by an application or network service. The services provide a method for multiple applications to communicate using the transaction models defined in the ANSI/ISA 95.05 and IEC 62264-5 standards. The ISBM:

- specifies the definition of services but does not define how the services are implemented
- specifies a general architecture for an ISBM implementation but does not define the architecture of any supporting applications or network services
- specifies the underlying external communication method but does not define any specific underlying internal communication methods

Multiple different implementations are envisioned. The network service will have to include some method for storage or caching of exchanged information, and must guarantee message delivery. However, the ISBM interface is designed to be independent of the underlying message transfer mechanism.

The ISBM essentially provides a standard interface to an ESB (Enterprise Service Bus) system or to any other message or file exchange system that offers guaranteed message and storage or caching of exchanged messages.

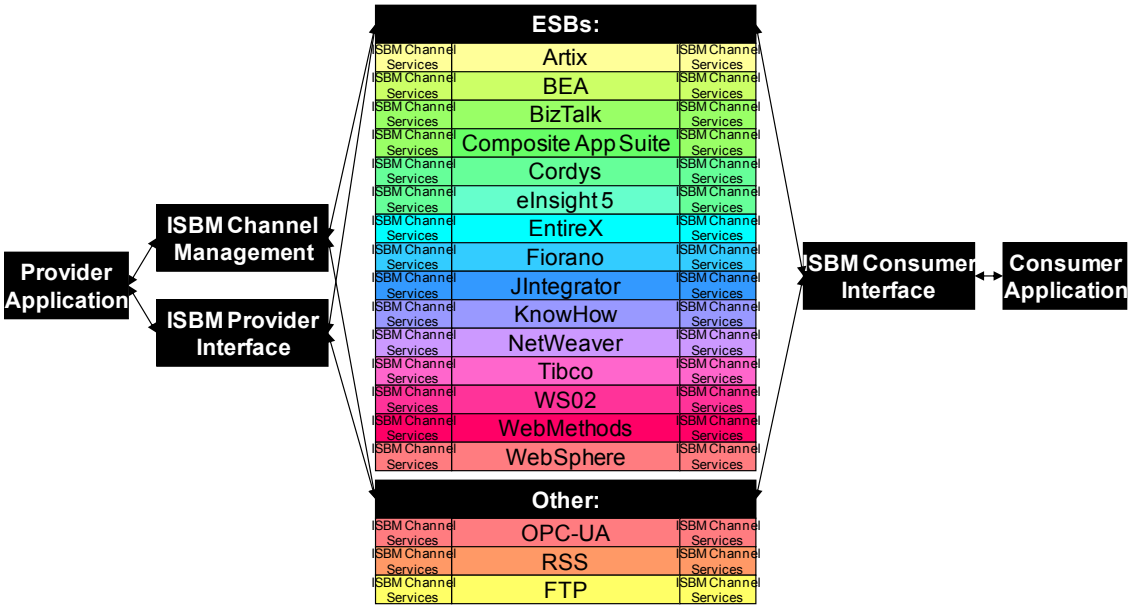


Figure 1 - ISBM Interface to ESB and Other Service Providers

Certain services are not defined by this specification, for example, quality of service, message validation, and transformation capability, but can be provided by the ISBM Service Provider to offer differentiation between suppliers and solutions.

2.2 Application to Application Data Exchange

Application to application data exchange is represented in the OSI communication model as a single “Application” layer. However, with the development of data object standards and data representation messages (such as B2MML, MIMOSA CCOM-ML, ISO 15926, OPC UA address

space Objects, and OAGIS Nouns), a simple single layer is insufficient to describe the complexity of object based, loosely coupled application-to-application transactional communication.

Three sublayers can be defined for the application layer for application-to-application communication: a data object layer, a transaction layer, and an exchange service layer, as shown in Figure 2. ISBM is a minimal interface subset that can reside on any exchange service layer and that is based on well-defined and structured data objects and transaction messages.

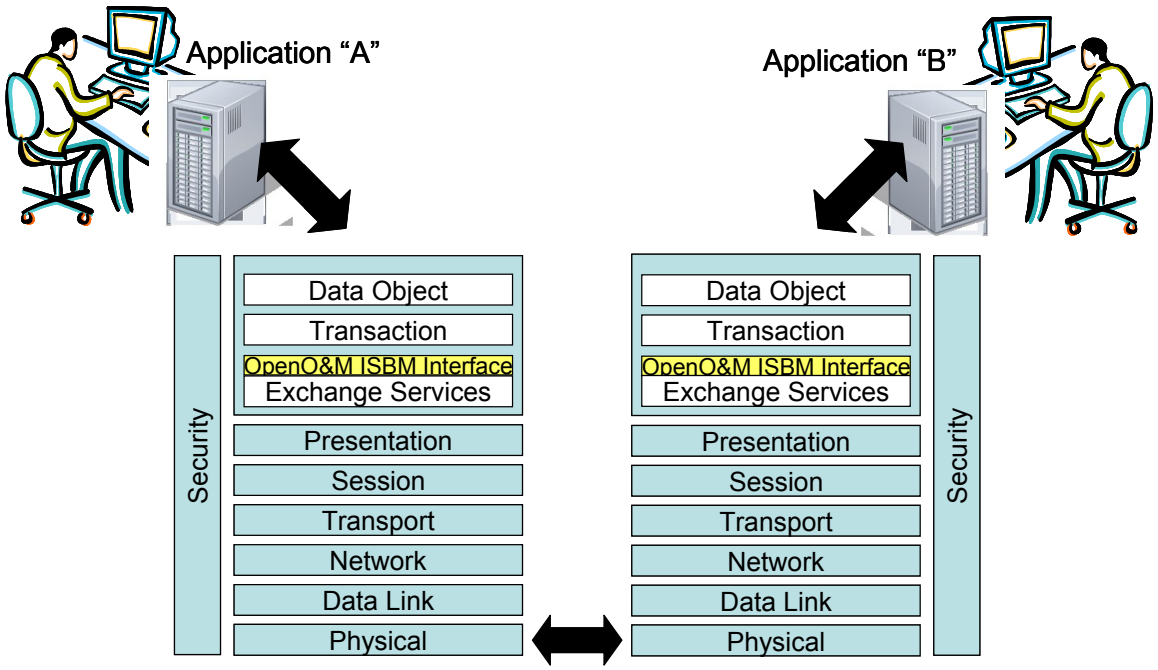


Figure 2 – Loosely Coupled Application Communication Stack

Each of these "Application" sublayers address a specific element of application data exchange, as shown in Figure 3:

1. The Data Object layer defines the meaning, format, and structure of the basic elements of exchanged information. This layer contains application space specific definitions, such as the ISA 95.02 object definitions, WBF B2MML, MIMOSA CCOM objects, ISO 15926 objects, OPC UA address space objects, and "Nouns" defined in OAGIS.
2. The Transaction layer defines the meaning, format, and structure of actions to be taken on the data objects. For the ISBM, this layer contains IEC 62264-5 transaction style specific definitions.
3. The ISBM Service Interface defines a minimal interface to the Exchange Service Layer.
4. The Exchange Services layer defines the meaning, format, and structure for coordination, buffering, and exchange of messages or files. This layer contains transfer or exchange style specific definitions, such as Enterprise Service Buses, Enterprise Message Delivery Systems, RSS, FTP, or Named Pipes.

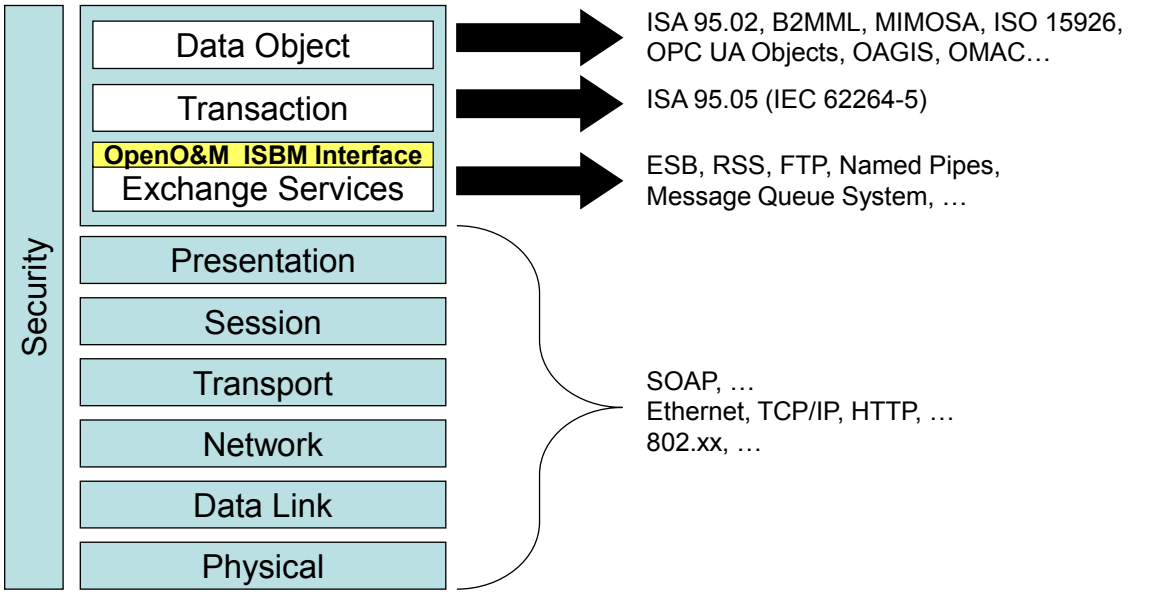


Figure 3 – Defined Standards at Each Application Sublevel

The OpenO&M Information Service Bus Model (ISBM) defines a set of transaction services that are suitable for use of exchange of OpenO&M information objects, using IEC 62264-5 transactions. In a sense, ISBM defines the standard “on-ramp” and "off-ramp" to a set of communication services. It defines how data is placed into exchange methods and how it is retrieved from the exchange methods.

2.3 Transaction Model

The ISA 95.05 and IEC 62264-5 standards define three models for business transactions: a publish model, a push model, and a pull model[1].

The ISBM defines a standard interface for applications to exchange data following any of the ISA 95.05 transaction models using OpenO&M XML schemas to represent data.

The transactions supported by the ISBM support:

1. A publish-subscribe model that supports multiple subscribers and multiple publishers, where the publishers and subscribers have not direct knowledge of each other.
2. A push and pull model, also called a request-response model, where an application sends unsolicited requests for a service and has no direct knowledge of the receiving application that will process the request and possibly return a response.

2.4 Communicating Applications

ISA 95 and IEC 62264 define four roles:

1. Information Provider (to receive GET messages and send SYNC messages)
2. Information Receiver (to receive PROCESS, CHANGE, and CANCEL messages)
3. Information Users (to send GET messages and receive SYNC messages)
4. Information Sender (to send PROCESS, CHANGE, and CANCEL messages).

[1] See the ISA 95 standards and WBF B2MML documentation for a complete description of the types, format and structure for transactions.

In the OpenO&M ISBM model these are simplified to Provider Application (Information Provider and Information Receiver) and Consumer Application (Information User and Information Sender), as shown in Figure 4.

An application can be a provider application, consumer application or both. If an application is both, then it should be a consumer of different data than it is provides.

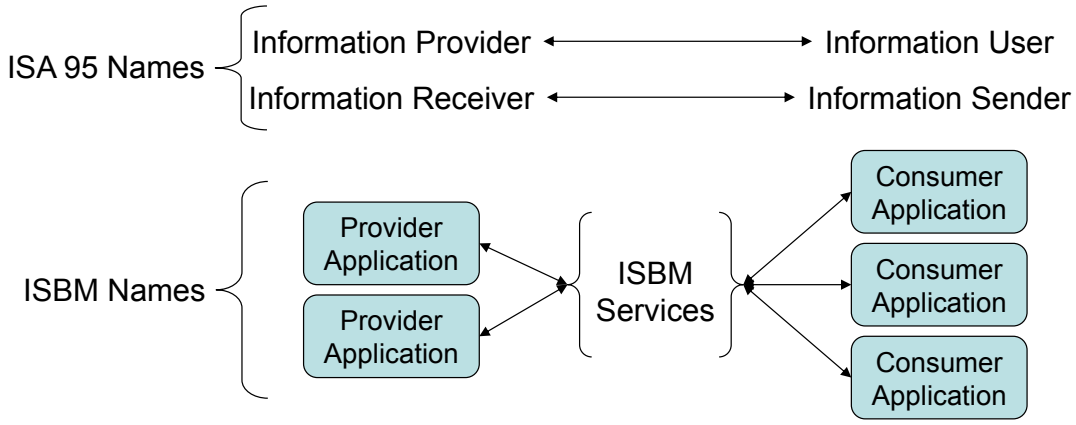


Figure 4 – OpenO&M Information Service Bus Model Names

2.5 Managed Communication Channels

The OpenO&M ISBM is based on the concept of managed communication channels. A "channel" is a software object that represents a specific many-to-many communication conduit between applications. Think of a channel as a channel in a CB radio, some channels are for requests and responses, some channels are for general information distribution. Channels have topics, think of a topic as a conversion topic within a CB channel, you can chose to listen to some topics on the channel but ignore others.

The assumption of the standard is that the ISBM services are provided by a communication application, applications, middleware, or services. The implementation method for the ISBM internal services are not defined here and multiple architectures are possible.

The ISBM provides a definition of the standard interfaces to the services (not how they are implemented).

- A managed communication channel is called a *Channel*.
- The services provided for each *Channel* are the *ISBM Channel Management Services*.
- A *Channel* is identified using a Channel URI. A Channel URI allows a hierarchy of channel definitions that match a company's physical or application structures, such as channels identified by plant site or major application suite name.
- An *ISBM Service Provider* is the application or network service that exposes and implements the *ISBM Channel Management Services*.
- A recommended structure for the *Channel* hierarchy is defined in this document.

Channels can support three general types of information exchange:

A. Publications – Information that may be sent to multiple consumer applications.
B. Requests – Information that may be sent to one or more provider applications.

C. Responses – Information returned from a request to a consumer application.

Each *Channel* supports two way communications between provider applications and consumer applications.

1. A *Channel* may be created to support one of the following: publication services (called a *Publication Channel*), request services (called a *Request Channel*), or response services (called a *Response Channel*).
2. A *Provider Application* may post publications to a *Publication Channel*.
3. *Consumer Applications* may subscribe to publication notifications (if notification services are supported by the implementation) and may read publications. If notifications are not supported, then the *Consumer Application* may poll the *Publication Channel* using the read publication service.
4. A *Consumer Application* may post requests to a *Request Channel*.
5. A *Provider Application* may subscribe to request notifications (if notification services are supported by the implementation) and may read requests. If notifications are not supported, then the *Provider Application* may poll the *Request Channel* using the read request service.
6. A *Consumer Application* may read responses to a request from a *Response Channel*.
7. A *Consumer Application* may subscribe to response notifications (if notification services are supported by the implementation) and may read responses. If notifications are not supported, then the *Consumer Application* may poll the *Response Channel* using the read response service.
8. *Channels* have associated *Topics*. Topics are identified when subscribing to a channel, when posting a publication, and when posting a request.

2.6 ISBM Channel Management Services

The ISBM Channel Management Services are shown in Figure 5. These services would usually be called used by a provider application, or by a dedicated channel management application when dealing with legacy applications.

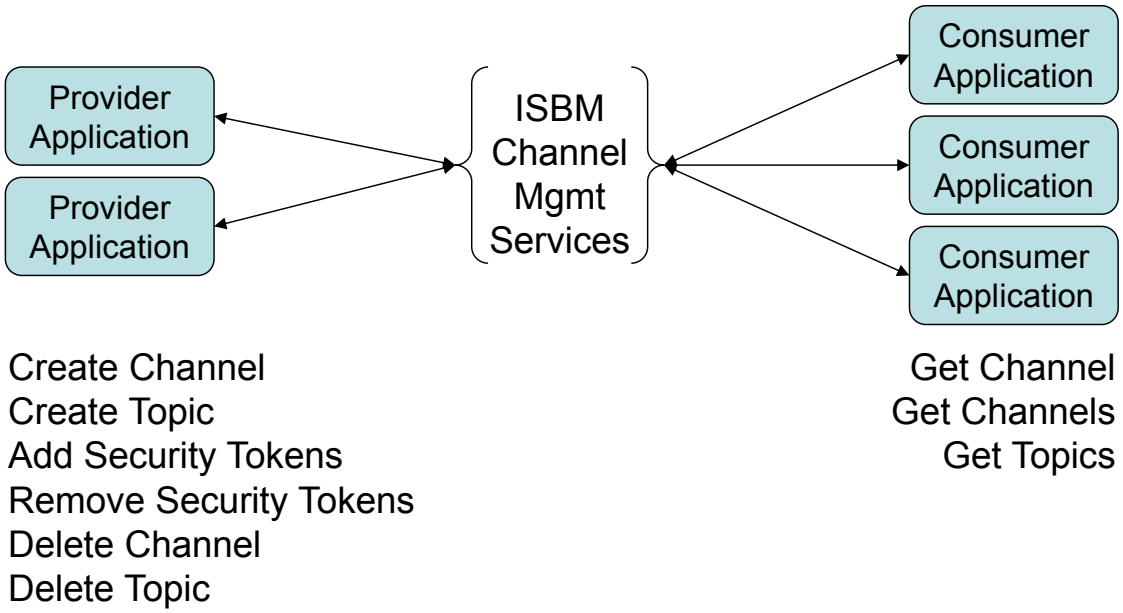


Figure 5 – ISBM Channel Management Services

The ISBM Channel Management Services are used to create and delete channels and topics, and to query channel data.

Instead of dynamic channels and topics being created at session time, channels and topics are created upfront in order to support the concept of "fixed" application communications (commonly found in industrial environments) and to provide a mechanism for permissions management in governing channels and topics.

2.7 ISBM Notification Services

The single NotifyListener service allows the ISBM Service Provider to indicate to a provider or consumer application that a message that meets their read criteria is waiting to be read. It provides an asynchronous callback alternative to the provider/consumer application polling the ISBM.

If a provider/consumer application does not provide a valid URI hosting the NotifyListener service, no notification will be provided to the respective application.

2.8 ISBM Publish-Subscribe Services

The ISBM Publish-Subscribe Services are shown in Figure 6. The services allow multiple provider applications to post publications to the same channel or different channels. Consumer applications may subscribe to callback notifications of a new message in their subscription or can poll to read publications. Topics and XPath expressions provide subscription filtering mechanisms.

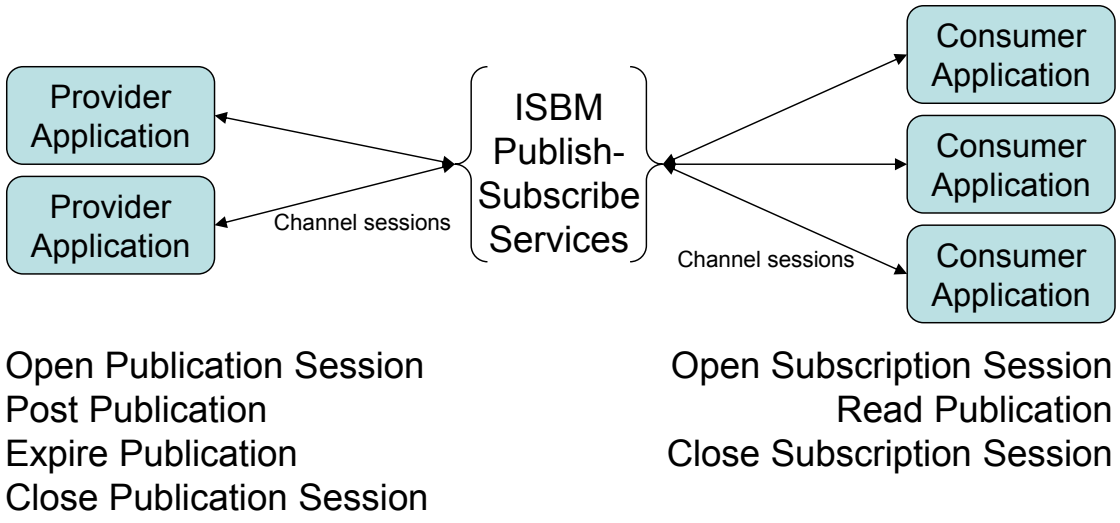


Figure 6 – ISBM Publish-Subscribe Services

2.8.1 Publish-Subscribe with Notification Example

A publish-subscribe scenario with a single provider application, notification services available, and a consumer application able to use notification services is shown in Figure 7. (Note: there typically will be multiple consumer applications receiving publications, but only one is shown in this example for simplicity.)

In this scenario, the provider application opens an ISBM publication session for a given channel[2]. When the provider application has determined that data should be published, it posts publication (a) with a message topic and no message expiry.

A consumer application subscribes to the ISBM publication channel using the same channel and list of topics. As the first publication occurred before the consumer application opened a subscription session, a notification is not provided. In order to see if there are any existing messages posted prior to the subscription session creation and to synchronize the message pointer for being notified as a listener, the consumer application attempts to continually read publications until it has read all messages.

When the second message (b) is sent by the provider application, the consumer application is notified that a new message now exists in its subscription, and the consumer application attempts to read the publication, passing (a) as the LastMessageId. Message (b) is consequently returned.

The consumer application closes its subscription session (e.g. scheduled maintenance), and subsequently opens a new session on the same channel and topic as before. In the meantime, the provider application has published the new message (c). No notification is provided but when the consumer application attempts to read a publication, passing (b) as the LastMessageId, message (c) is returned.

[2] It is assumed that the appropriate channels and topics have been created prior to the scenario.

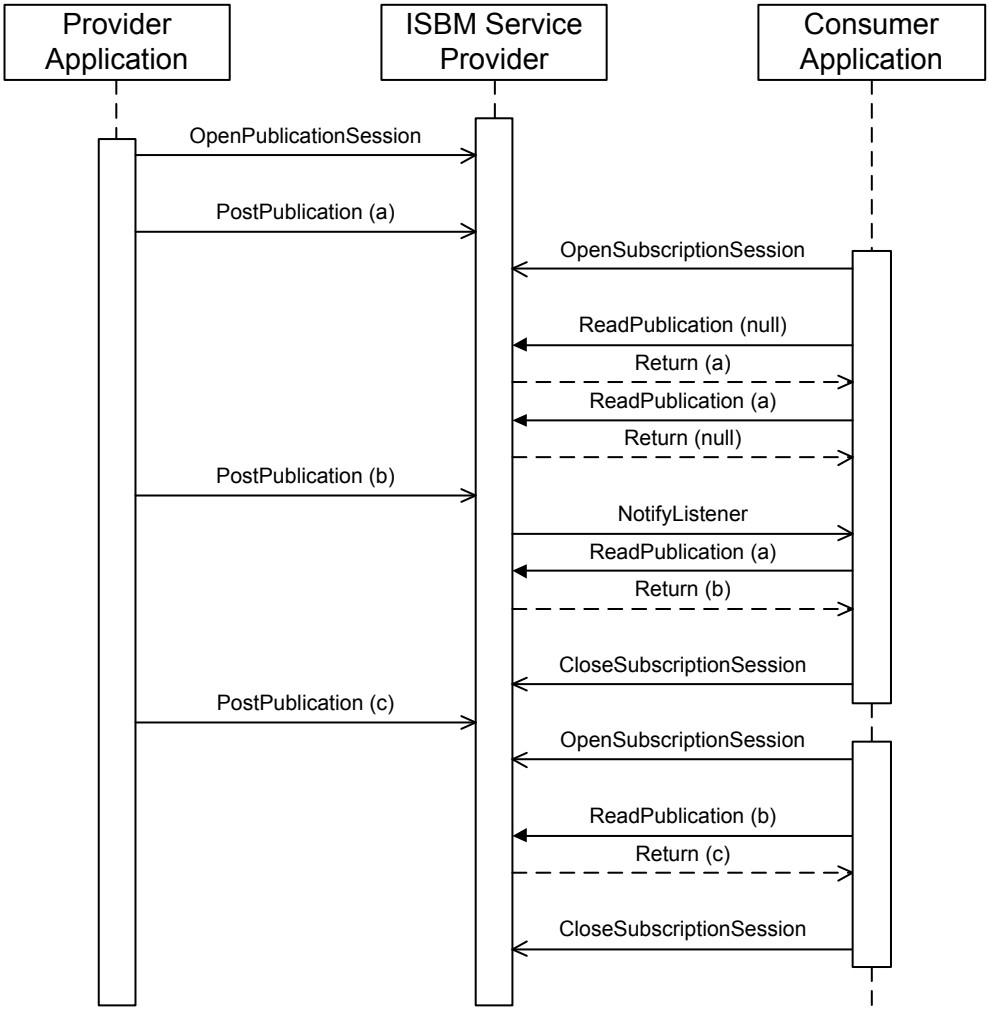


Figure 7 – Publish-subscribe scenario with notification

2.8.2 Publish-Subscribe without Notification Example

A publish-subscribe scenario with a single provider application, where notification services are not available or the consumer application is not able to use notification services is shown in Figure 8. In this scenario, there is no change for the actions of the provider application as in the previous scenario.

In this scenario the consumer application polls the ISBM channel for publications either periodically or based on some local event. The returned information from the ReadPublication indicates if a new publication was returned. In the event where (null) is returned, the next ReadPublication call uses the same LastMessageId as the previous call.

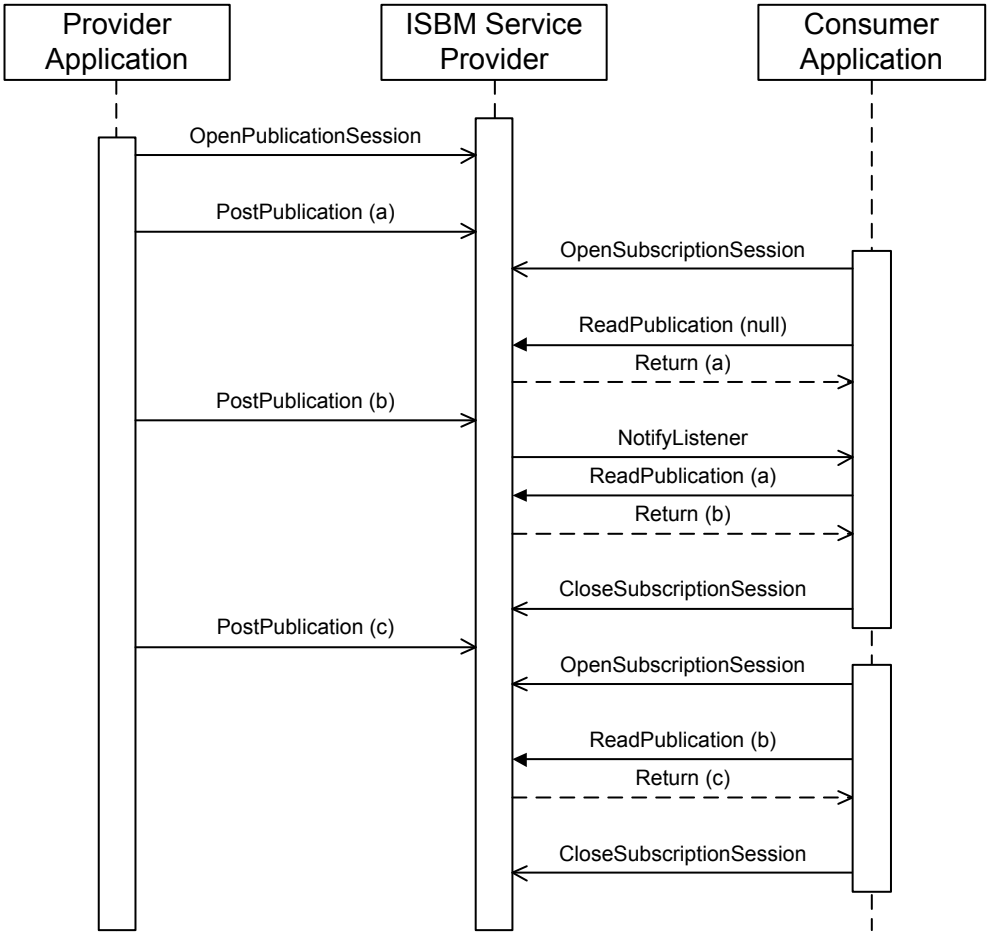


Figure 8 – Publish-subscribe scenario without notification

2.8.3 Publish-Subscribe with Multiple Providers Example

More than one provider application may use the same publication channel. The scenario shown in Figure 9 has two provider applications. The example is similar to the Publish-Subscribe with Notification example, and although the notifications for messages (b) and (c) arrive at the consumer application almost simultaneously, the consumer application's behavior does not change.

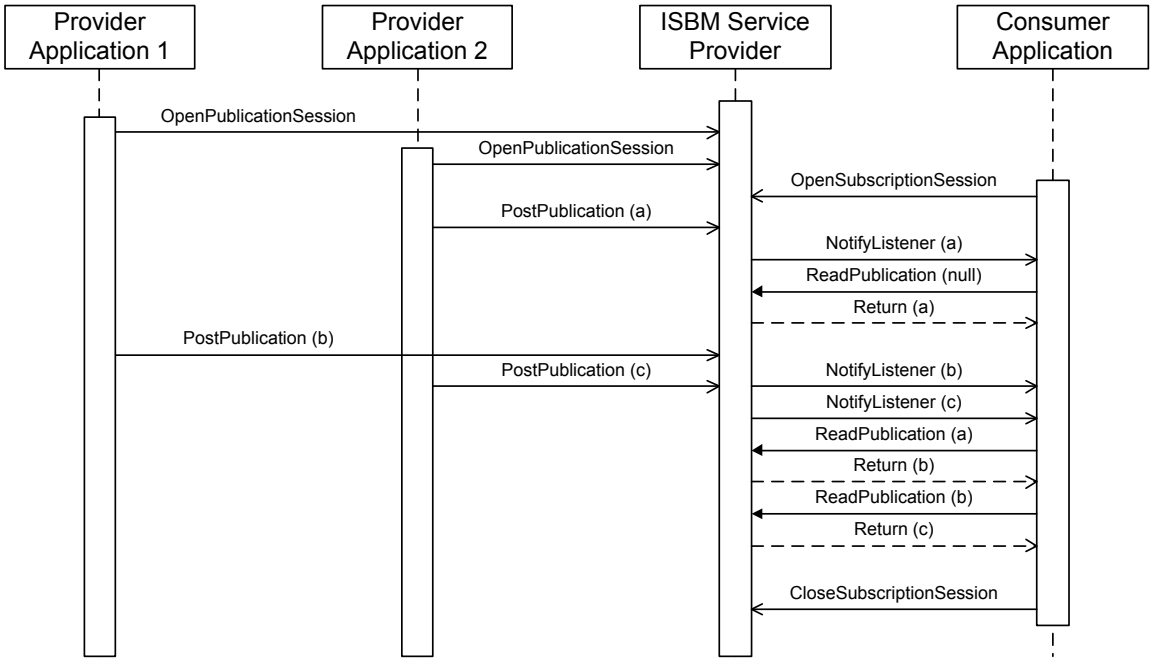


Figure 9 – Publish-subscribe scenario with multiple provider applications

2.8.4 Publish-Subscribe with Expiry Example

Message expiry can be used by provider applications to remove messages from a consumer application's visibility. This could be due to a number of reasons, including changing relevancy of the message or inaccuracies in the message[3]. The scenario in Figure 10 highlights expiry behavior in two cases: where a read message has expired and where an unread message has expired. The second ReadPublication call by the consumer application uses a LastMessageId of (a), and despite message (a) being expired, the ISBM Service Provider returns the next message in the subscription – although in this case, there is no message. The third ReadPublication call still uses the LastMessageId of (a), and as message (c) has since expired, message (d) is returned.

[3] As no notification of expiry is provided to consumer applications, revocations should be conducted by sending subsequent messages.

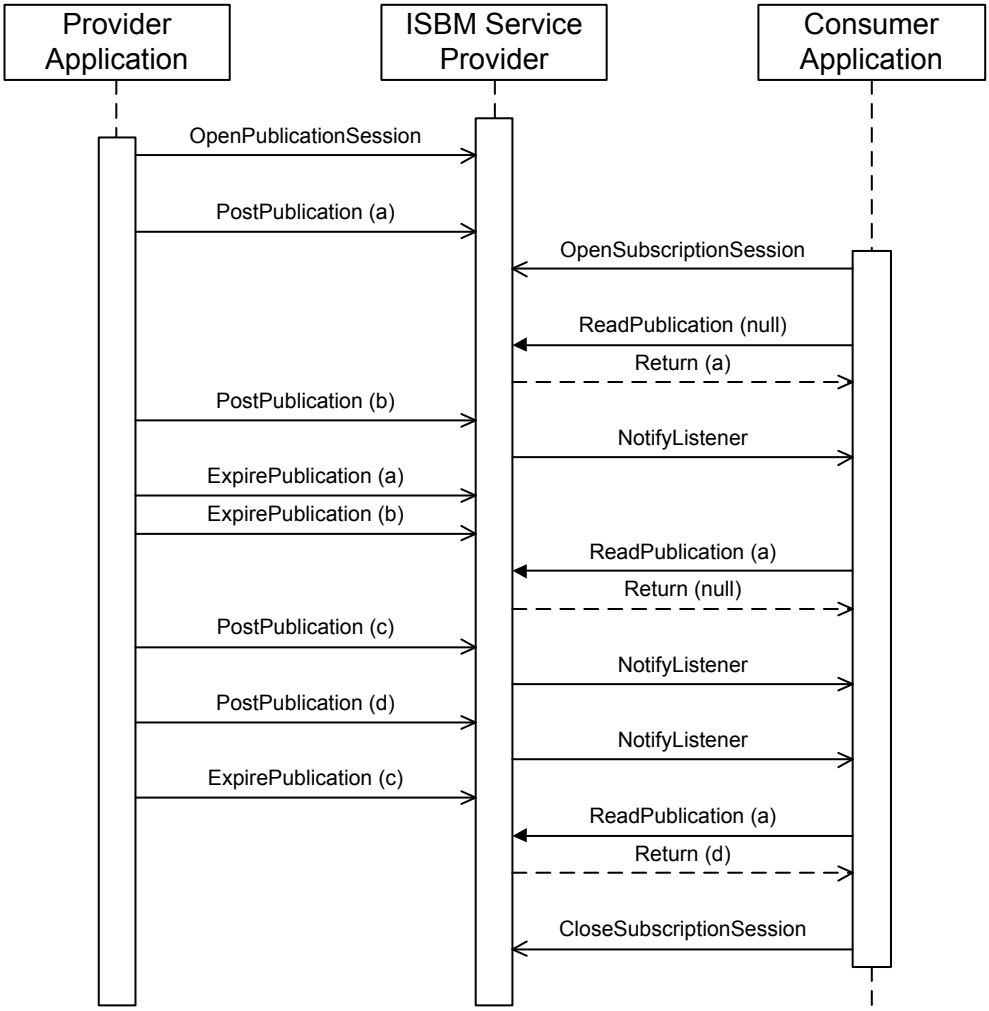
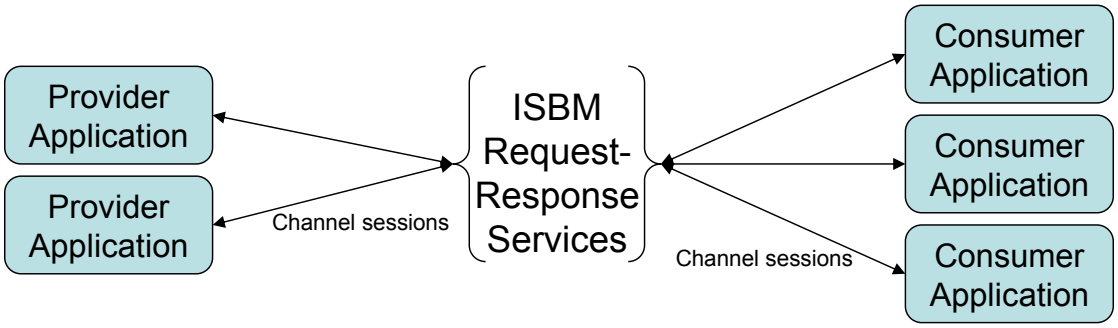


Figure 10 – Publish-subscribe scenario with expiry

2.9 ISBM Request and Response Channel Services

The ISBM Request-Response Channel Services are shown in Figure 11. The services allow one or more Consumer Applications to post requests to Provider Applications, allow one or more Provider Applications to read requests and post responses, and for the Consumer Application to read the response. Topics allow Provider Applications to determine if it should process the request and post a response to the requestor.



Open Read Request Session
Read Request
Remove Request
Close Read Request Session
Open Post Response Session
Post Response
Close Post Response Session

Open Post Request Session
Post Request
Close Post Request Session
Open Read Response Session
Read Response
Remove Response
Close Read Response Session

Figure 11 – Services for Request/Response

2.9.1 Request-Response with Notification Example

A request-response scenario with a single provider application, notification services available, and a consumer application able to use notification services is shown in Figure 12.

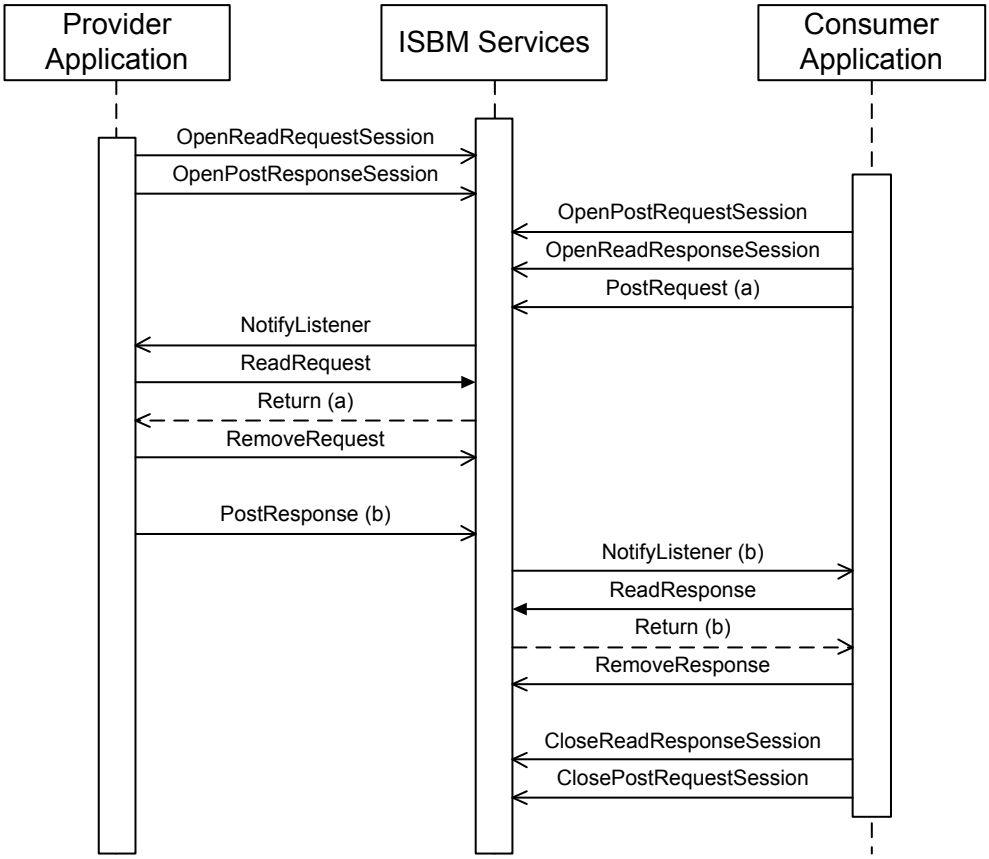


Figure 12 – Request-Response scenario with notification

In this scenario, the provider application subscribes to the request channel. A consumer application opens the request channel and posts a request. The provider is notified and reads the request. The provider application performs its appropriate function (in this case to get data) and sends the response message. The consumer application is notified of the posting and reads the request. While not shown in the scenario, a provider application may post multiple responses depending on the scenario, in which case the consumer would receive multiple notifications.

2.9.2 Request-Response without Notification Example

If the applications or ISBM services do not support notification, then the provider and consumer applications may poll for a request or response. Figure 13 illustrates a request-response scenario where the consumer application must poll for a response.

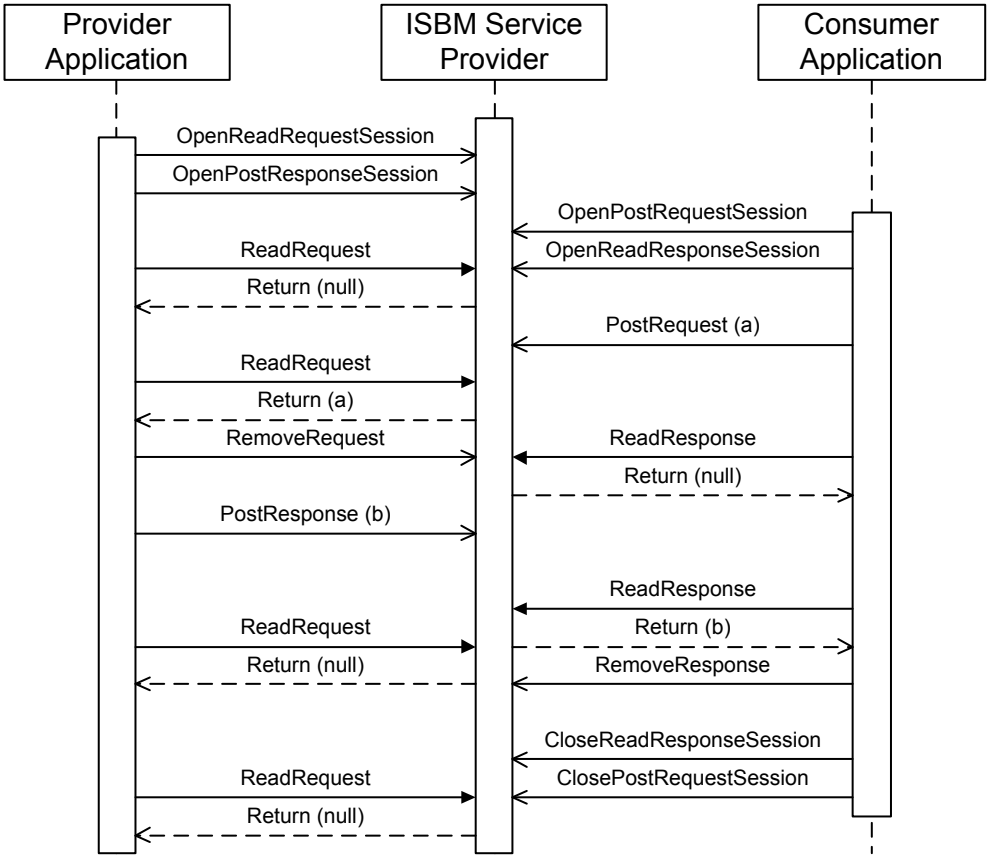


Figure 13 – Request-Response scenario without notification

2.9.3 Request-Response with Multiple Providers Example

Figure 14 illustrates a scenario with multiple provider applications. In this case two provider applications have subscribed to requests on the same ISBM channel. The consumer application posts a request with a specific topic (such as Personnel Information).

Provider Application 1 is notified of a request that matches a topic that it subscribed to. Provider Application 1 reads the message and generates a response. Provider Application 2 is not notified of the request, because the topic does not match a subscribed topic.

In this scenario, the consumer application is not able to handle notifications, so it polls the ISBM services for a response message.

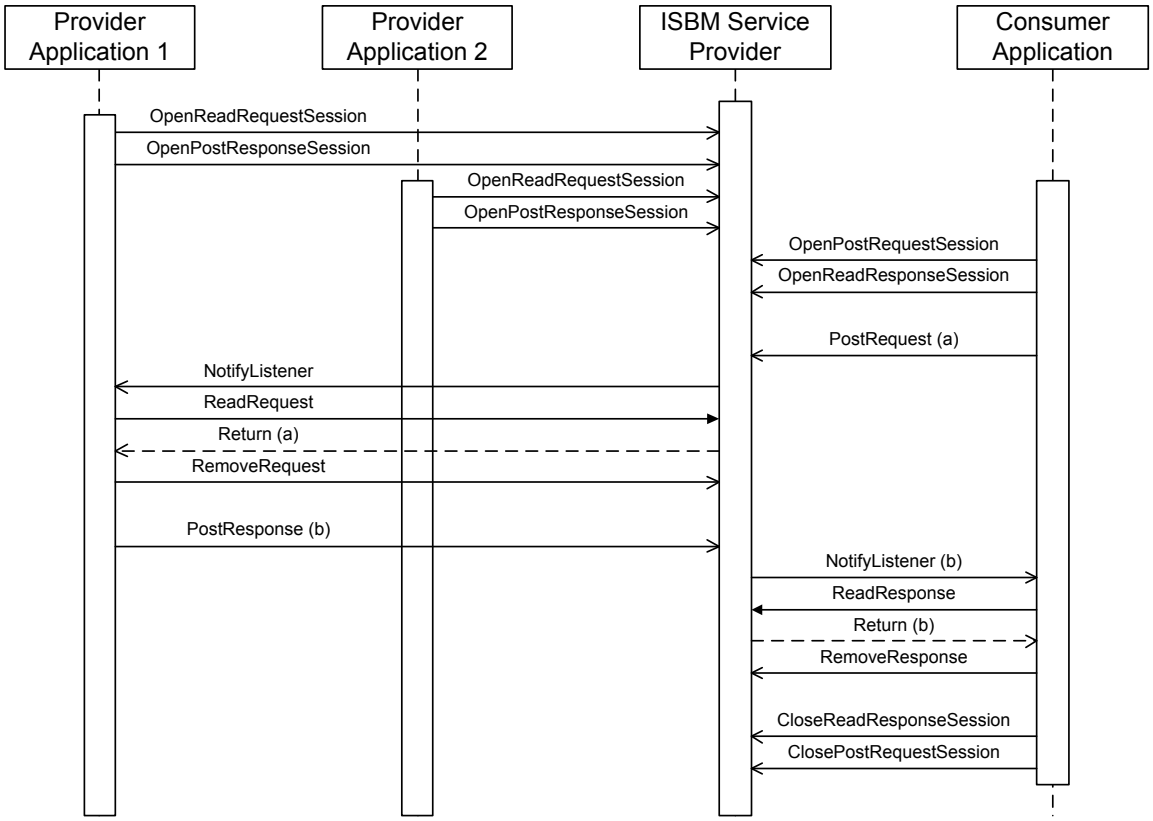


Figure 14 – Request-response scenario with multiple provider applications

Note: A full system should not have multiple providers for the same topic on the same request channel. If this occurs then there is a possibility of an indeterminate number of response messages that would be returned to the consumer application. This consideration requires careful design of a system of applications to remove dual responsibility for request topic provider applications.

3 ISBM Technical Requirements

3.1 Channel URIs

Channel URIs should be defined as a name hierarchy determined by the company or the application suites. Channel URIs should follow the syntax:

/<ISBM root>/<channel scope>/<information scope>/<channel use>

For example:

/AJAXEnterprises/Company/Material/Checkpoint
/AJAXEnterprises/Company/Material/Request
/AJAXEnterprises/Company/Material/Response
/SystemTest/Final/OurMaterialManager/Inventory/Changes
/AJAXEnterprises/France/Personnel/Checkpoint

3.1.1 ISBM Root

The ISBM Root is the root of a hierarchy defined when the ISBM services are installed or initialized. Depending on the ISBM Service implementation there may be one or more roots allowed. The ISBM is used to define the top level of the channel hierarchy when browsing the hierarchy. The ISBM Service Provider may require specific values for ISBM Root.

For example:

1. AN ISBM root may be the name of the company, such as: "*AJAX*" or "*AJAXEnterprises / SpecialToolCo*".
2. AN ISBM root may be a related set of services, with sets for testing, deployment, and operations, such as: "*SystemTest / Beta*", "*SystemTest / Final*", "*SpecialToolCo / Operations*".

3.1.2 Channel Scope

The channel scope contains a hierarchy that may correspond to a physical, geographical, or logical grouping determined by the enterprise, application or project. It may be used to limit the scope of the exchanged information, such as information only exchanged within a one division of a company. The hierarchy may include site, region, division, area, software system or any other enterprise defined element.

For example:

1. A channel scope may include a site or region name to limit the number of distributed messages, such as: "*AsiaPacific*", "*SouthAfrica*", or "*France*".
2. A channel scope may be a software system, because the information is provided by a well-known system name, such as "*OurMaterialManager*", "*PersonnelTracker*", "*InventoryDatabase*".
3. A channel scope may be companywide because the information is intended for any application in the company. In this case the channel scope should indicate the entire enterprise or company, such as "*Enterprise*" or "*Company*", or it may be null.

3.1.3 Information Scope

The information scope defines the range or general type of information exchanged. The information scope may be related to transaction nouns, to other collections of objects, or to business or control processes that deal with a collection of objects.

For example:

1. An application that handles all forms of material information may define a channel with an information scope of "*Material*".
2. An application that only handles Material Lot and Sublot inventories may define a channel with an information scope of "*Inventory*".

3.1.4 Channel Use

The channel use qualifies the information scope to indicate how the information is being used. The channel use may be related to transaction verbs or other business or control process that deal with how the information on the channel is to be used.

For example:

1. An application that sends material requests may define a channel with a channel use of "*Request*".
2. An application that indicates changes handles Material Lot and Sublot inventories may define a channel with a channel use of "*Changes*".

3.2 Topics

Topics are used in application services to limit or filter the type of information that is obtained from read and notify requests for Provider Applications and Consumer Applications.

Topics are also used by Provider Applications to specify the type of information that they will be publishing or posting on an ISBM *Channel*.

Topics allow a single channel to handle a collection of different types of data, yet still provide a method for the receiver of the data to limit the types of data that it is required to handle.

The same topic may be defined on multiple channels. For example:

1. There may be a *ProductionSchedule* topic defined for *CheckPoint* and *Changes* channels with a site channel scope, and a *ProductionSchedule* topic defined for *Checkpoint* and *Changes* channels for an area channel scope.
2. There may be a *QualificationTest* topic defined for a *Request* channel at the enterprise channel scope, and a *QualificationTest* topic defined for a *Request* channel at the country channel scope.

3.3 XPath Filtering

To allow efficient content filtering of messages, an XPath expression can be optionally added to a subscription or read request session to provide a content filtering definition. The XPath expression must be defined as an XPath v1.0 expression that returns a node set that is considered valid XML. An XPath evaluation that returns an empty node set will not cause a notification to be generated nor will be visible as a message to the receiving system.

As valid XML has a single root element, an XPath evaluation that returns multiple nodes is considered invalid and the behavior is the same as that of an empty node set in that no notifications will be generated and the message will not be visible to the receiving system. For an XPath expression that use namespaces, multiple namespace prefixes and names can be added upon session creation.

3.4 Publication Expiration

A publication can be flagged as expired by a provider application via the ExpirePublication method or via a time-based expiry. With the time-based expiry, the expiry time is calculated based on the *completion* of invocation of the PostPublication method plus the specified duration. Despite a time-based expiry duration being specified, the publication can be expired via the ExpirePublication method prior to its time-based expiry.

Expired publications are no longer available to subscribing consumer applications nor accessible to the provider application.

3.5 Security

Security in the ISBM services is of paramount importance. When using the ISBM, the communication applications have no knowledge of their communication partners, and do not know if there are none (for a publisher with no subscriptions), one, or many. Therefore, security cannot be defined as communication with trusted partners, instead security is defined as communication through secure channels.

3.5.1 Security Tokens on Channels

Security is managed through security tokens. Security tokens are assigned to channels by the provider applications. The security tokens are used by applications when opening a session. If the application provided security token does not match a security token assigned to the channel, then no channel information is returned. The security tokens assigned to a channel is a unique list – there are no duplicates.

Security tokens are exchanged in an out-of-band communication channel, such as manual exchange of tokens, or electronic exchange through a secure point-to-point channel.

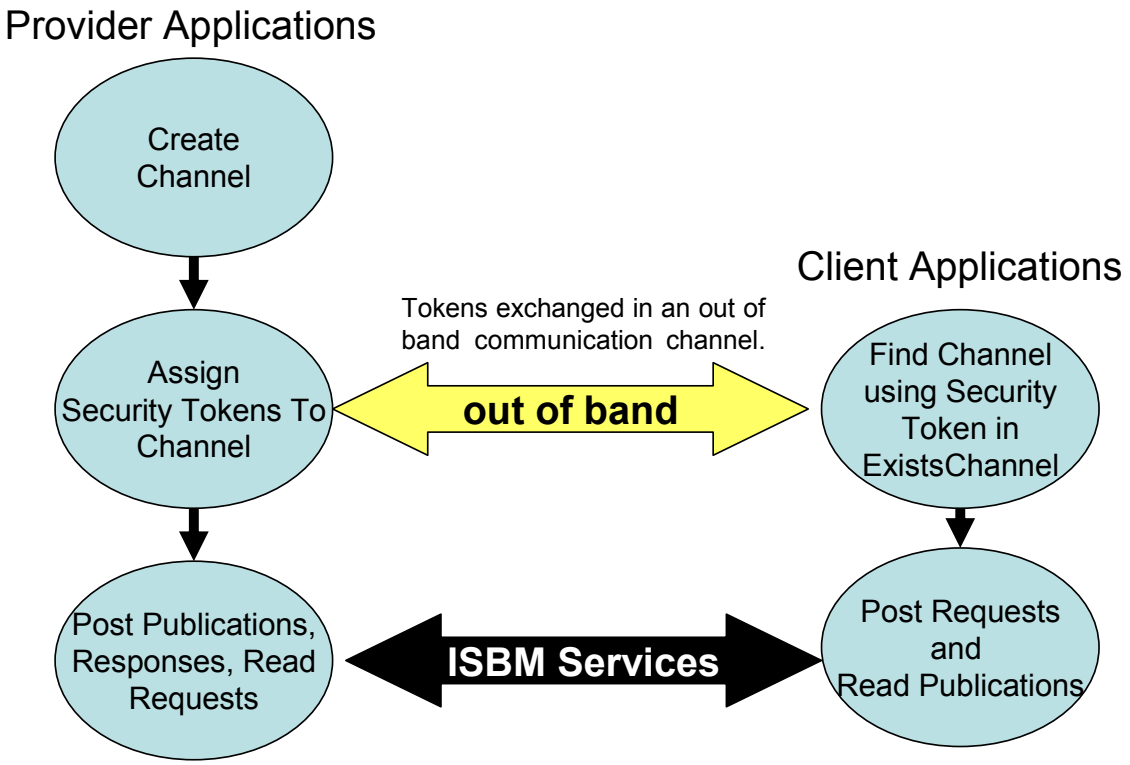


Figure 15 – Security of Channels

3.5.2 Security Token Format

Security tokens must follow the WS-Security standard. There are a large number of ways to validate a user using WS-Security. This specification defines three formats for security tokens.

1. Username/Password
2. PKI through X.509 Certificates
3. Kerberos

ISBM Service Providers must support, at a minimum, the three aforementioned security token formats; *ISBM Service Providers* may provide additional formats for security tokens. In this case the *ISBM Service Provider* must supply or make available an appropriate Security Token Service to create and acquire security tokens.

Tokens are XML documents that follow the WS-Security definition for a SOAP Header element to carry security-related data.

The specific security token specification followed by ISBM is defined in:

http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

The XML schema that defines the format for token representation is:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd

This specification defines the <wsse:Security> header as a mechanism for conveying security information with and about a SOAP message. This header is, by design, extensible to support many types of security information.

For security tokens based on XML, the extensibility of the <wsse:Security> header allows for these security tokens to be directly inserted into the header.

3.5.2.1 Username/Password

A common way to identify security is through the use of a username and password combination. WS-Security has defined the UsernameToken element to pass user credentials in this manner. The schema definition for this element is:

```
<xs:element name="UsernameToken">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="Username"/>
            <xs:element ref="Password" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="Id" type="xs:ID"/>
        <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
</xs:element>
```

This schema element references the Username and Password types. These two types are strings that contain extra attributes as needed.

The Password element contains an attribute named Type that indicates how the password is being passed around. An example UsernameToken with a password that is encrypted is:

```
<UsernameToken>
  <Username>Bob Smith</Username>
  <Password Type="PasswordDigest">
    KE6QugOpkPyT3Eo0SEgT30W4Keg=
  </Password>
  <Nonce>5uW4ABku/m6/S5rnE+L7vg==</Nonce>
  <Created xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
    2002-08-19T00:44:02Z
  </Created>
</UsernameToken>
```

3.5.2.2 PKI Through X.509 Certificates

Security tokens may specify an X.509 certificate. X.509 is an ITU-T standard for a public key infrastructure (PKI) for single sign-on (SSO) and Privilege Management Infrastructure (PMI).

When a message sends an X.509 certificate, it passes the public version of the certificate in a WS-Security token named BinarySecurityToken. The certificate is sent as base64 encoded data. The BinarySecurityToken has the following schema:

```
<xs:element name="BinarySecurityToken">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Id" type="xs:ID"/>
        <xs:attribute name="ValueType" type="xs:QName"/>
        <xs:attribute name="EncodingType" type="xs:QName"/>
        <xs:anyAttribute namespace="##other"
                         processContents="strict"/>
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

```
</xs:element>
```

3.5.2.3 Kerberos

Kerberos is a computer network authentication protocol, which allows nodes communicating over a non-secure network to prove their identity to one another in a secure manner. It is also a suite of free software published by Massachusetts Institute of Technology (MIT) that implements this protocol. Its designers aimed primarily at a client-server model, and it provides mutual authentication – both the user and the server verify each other's identity.

The Kerberos specification used in ISBM is defined in:

http://www.ws-i.org/Profiles/KerberosTokenProfile-1.0.html

Message layer security with the Kerberos protocol in WSE 3.0 involves the following participants:

Client: The client accesses the Web service. The client provides the credentials for authentication during the request to the Web service.

Service: The service is the Web service that requires authentication of a client prior to authorizing the client.

Key Distribution Center (KDC): The KDC is the broker that authenticates clients and issues service tickets.

The main steps in the client side of a Kerberos system is:

1. Request a service ticket from the KDC.
2. Retrieve the service ticket from the KDC.
3. Send the message to the service using the service ticket as the security token.

The service authenticates the client using information found in the security token. The main service side steps are:

1. Validate the token.
2. Verify the XML signature.
3. Perform the specified service.
4. Initialize and send a response to the client (optional).

A Kerberos token may contain the schema used to validate the token, must defined a value type of the token (as defined by the KDC), the encoding type of the token, and the token. Some examples of Kerberos tokens are:

```
<wsse:BinarySecurityToken
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/xx/secext"
  wsu:Id="myToken"
  ValueType="wsse:Kerberosv5ST"
  EncodingType="wsse:Base64Binary">
    MIIEZzCCA9CgAwIBAgIQEmtJZc0...
</wsse:BinarySecurityToken>
```

```
<wsse:BinarySecurityToken
  wsu:Id="myKerberosToken"
  ValueType="http://docs.oasis-open.org/wss/2005/xx/oasis-2005xx-wss-
kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ"
  EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
    YIIEZzCCA9CgAwIBAgIQEmtJZc0...
```

```
</wsse:BinarySecurityToken>
```

3.5.3 ISBM Service Provider Implementations

1. All *ISBM Service Providers* **must** implement the three forms of security tokens.
2. The form, format, and out-of-band token exchange **must** be defined by the *ISBM Service Provider*.
3. *ISBM Service Providers* may choose to limit the ability to use the ISBM Channel Management services to approved applications, servers, or domains in order to increase security.
4. Channel management applications may chose not to apply security tokens to channels. While there is a requirement that the *ISBM Service Provider* implements security services, there is no requirement that a specific implementation use the services.

   For example:

   - A system may share information across companies through open Internet channels. In this case an *ISBM Service Provider* implementation should provide a strong security token system through a public key mechanism with specific security token assigned to specific communicating companies.
   - A system may be entirely contained within a secure environment behind both corporate and operations firewalls. In this case the user may decide to not assign security tokens to channels.

3.5.4 ISBM Application Implementation Considerations

An ISBM application implementation should take the following concerns and issues into account:

1. Security tokens will usually be stored in the provider and client applications so they can be used on startup or restart of the application. The tokens should be saved in a secure manner to prevent unauthorized discovery of the tokens.
2. In high security environments there may be a unique security token assigned for each possible communication path and security tokens may be changed on a regular basis.

4 ISBM Service Provider Considerations

The following sections define ESB type services that **can** be provided by *ISBM Service Providers*. The services are **not** part of the ISBM specification, but provide some of the areas in which vendors and others can provide differentiated service.

4.1 Security Considerations

An *ISBM Service Provider* should take the following concerns and issues into account:

1. The *ISBM Service Provider* may store messages in a persistent data store. If this is the case and there is security on the channel, then the stored messages may need to be encrypted to prevent unauthorized access to the stored messages.
2. Requests for access with invalid security tokens should be logged. They either indicate a problem with configuration information or a possible attack of the system.
3. Requests for access to invalid channel URIs should be logged. They either indicate a problem with configuration information or a possible attack of the system.
4. Messages exchanged within the ISBM Service implementation may require encryption or connection through secure channels. The method used may be dependent on the transport services used and is not defined in the ISBM interface.

5. Session IDs should be globally unique and use restricted to a specific provider or consumer in order to prevent access to a channel without going through token security.

4.2 Notification

ISBM Service Providers are encouraged to implement notification capability utilizing the provided notification service. This specification also allows light weight *ISBM Service Provider* implementations, where polling is an acceptable method for synchronization of applications.

4.3 Data Format Validation

ISBM Service Providers could provide data format validation services for messages. If the message are to follow a predefined and well specified format, such as B2MML or BatchML, then the service provider could provide a service to check the syntax correctness of posted messages. This would provide a governance check on messages. This could be implemented by the ISBM Service Provider maintaining a map between topic namespaces and XML Schema files.

4.4 Allowed Application Checking

ISBM Service Providers could provide a governance check that applications creating and subscribing to channels are allowed applications. This check would provide an additional level of security, which may be important if the ISBM Services go outside the company.

4.5 Data Exchange Logging

ISBM Service Providers could provide services to log all or selected messages for purposes of governance, compliance, and auditing. Because all messages are in an XML format, and the posting application is know, this could provide an audit or error tracing log that captures all in-band communications.

4.6 Common Error Handling

ISBM Service Providers could provide services for a consistent method for handling errors detected by provider and consumer applications. An error handling service, provided as a dedicated channel, could be used to determine the response to the error. Depending on the error, such as; invalid message received, lost message, incorrect data in message, or failure in ISBM services, the error handling service could notify the appropriate person or entity with responsibility.

4.7 Data Transformation Services

ISBM Service Providers could provide transformation services for messages. Typically this would be from a provider or consumer application specific format into a common format (such as B2MML or BatchML), and from a standard format to an application specific format.

A possible method to handle the transformation interfaces is through topics. Topics may be defined that match the application specific format for the messages. The *ISBM Service Provider* could provide a method for associating a topic to a transformation mapping. When a message is received with a transformation topic, then the *ISBM Service Provider* would transform the message to a standard format. When a read request is received with a transformation topic, then the *ISBM Service Provider* would transform the standard format into the application specific topic format.

The *ISBM Service Provider* would maintain the relationship between the application specific topics, the transformation rules to a standard, and a “standard” topic definition. There are no *ISBM Channel Services* for transformation. The assumption is that the transformation is not

handled by the applications, and that creating and maintaining the transformation rules and associations is handled by the *ISBM Service Provider*.

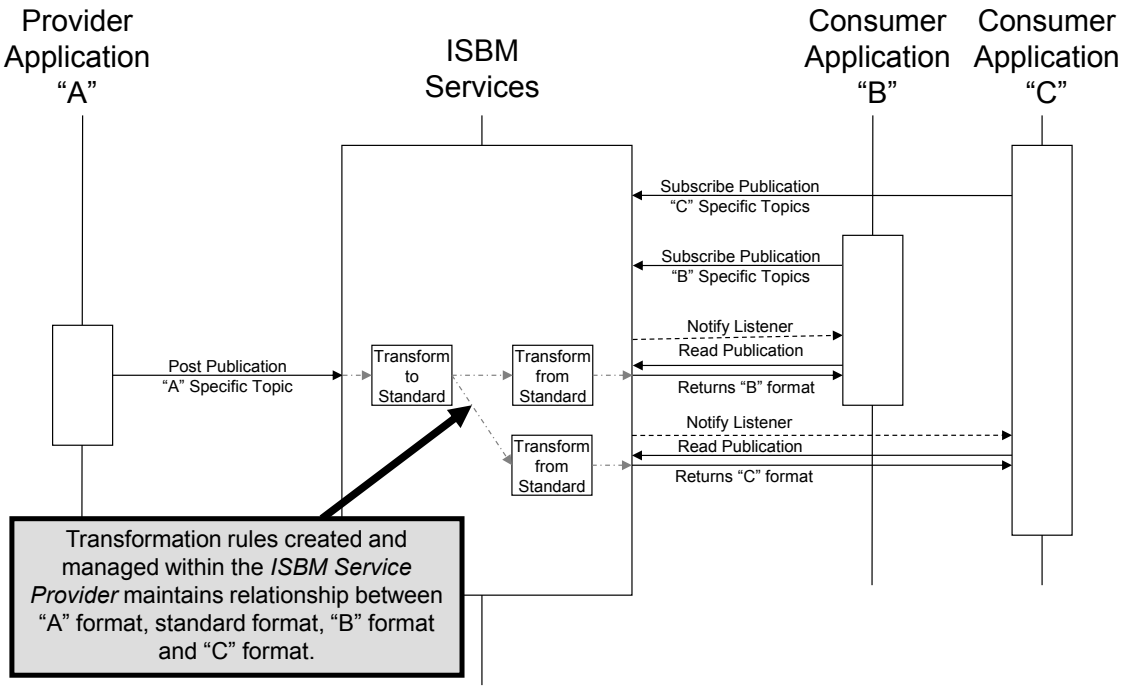


Figure 16 – Transformation Services with the ISBM Service Provider

4.8 Cross Company Bridge

ISBM Service Providers could provide cross company communication and authentication services for messages.

A method to provide chain of custody for published messages is shown in Figure 17. In this scenario a proxy application (or part of the ISBM) in Company A's environment would listen for publications from the ISBM. The proxy would forward the publications using a authenticated or secure method to a proxy application in Company B's environment. The receiving proxy would publish the message in Company B's ISBM environment. The bridge may also convert Channel and Topics from Company A's namespace to Company B's namespace.

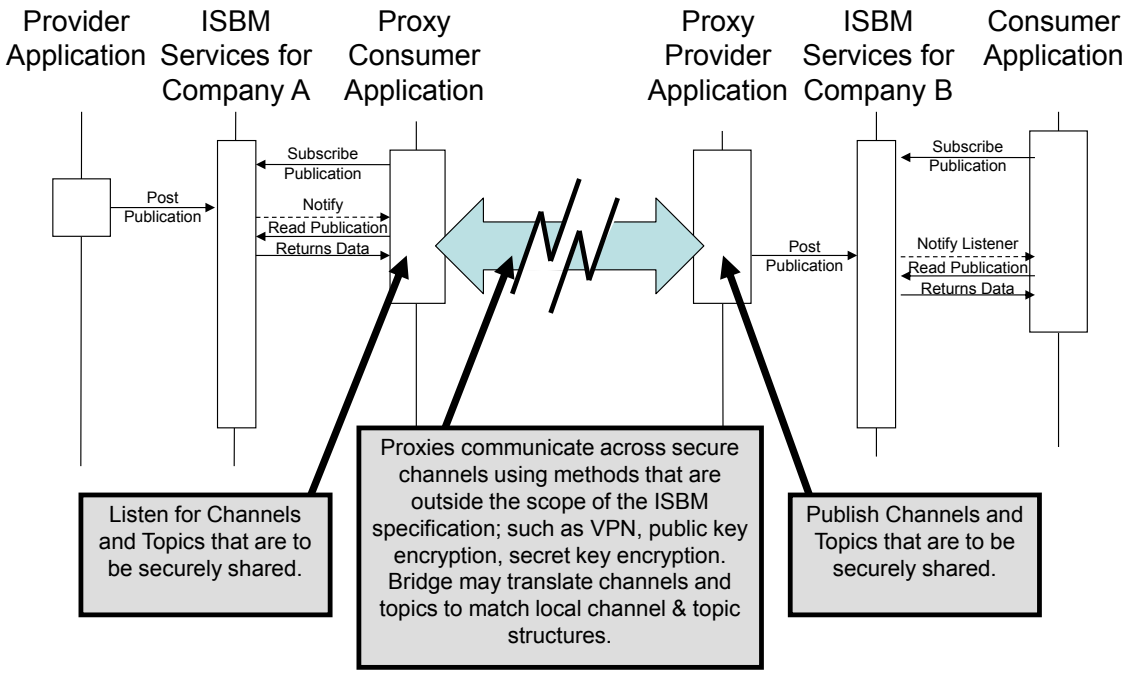


Figure 17 – Cross Company Bridge between multiple ISBMs

5 Service Definitions

This section defines the detailed format for the ISBM Service definitions.

5.1 Data Model

To assist the reader in understanding the data elements and relationships used by the ISBM services, a logical data model and data definitions are presented below. The data model is not prescribed as an implementation data model.

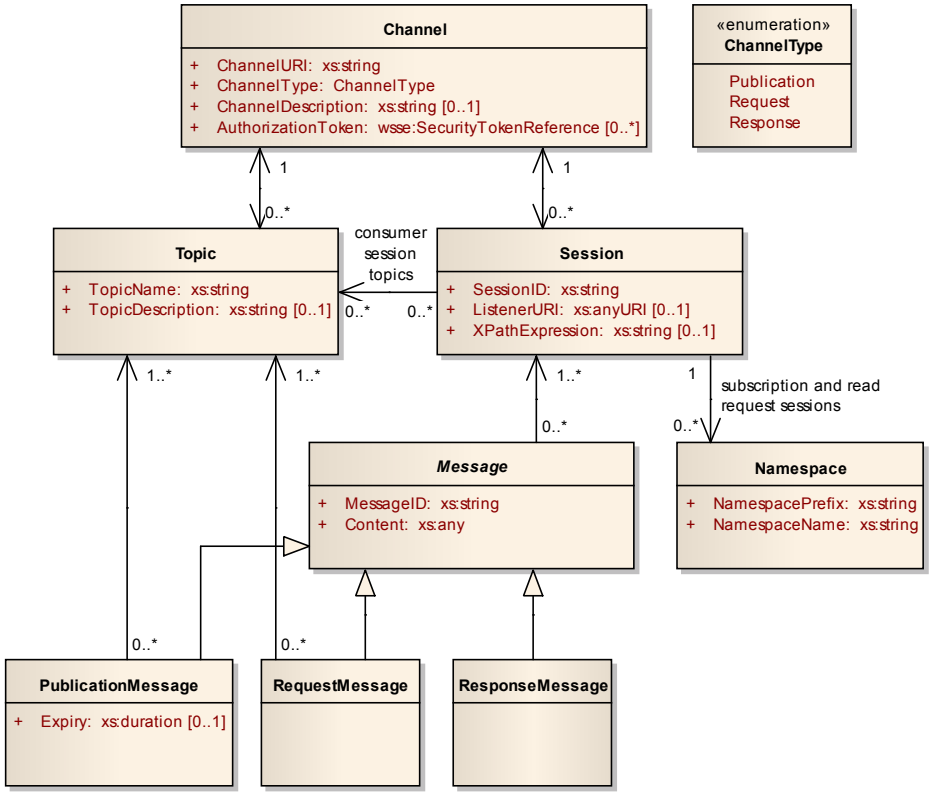


Figure 18 – Data Model

5.1.1 Data Dictionary

Entity	Attribute	Description
Channel	ChannelURI	The primary identifier for a channel. See Section 3.1 for details on the format.
	ChannelType	Indicates whether the channel is for publications, requests or responses. The ISBM can use the channel type to ensure the correct session creation service is called for a channel. Defined ChannelTypes are “Publication”, “Request” and “Response”.
	ChannelDescription	The description of a channel.

	AuthorizationToken	The security tokens assigned to the channel. Must be a distinct set.
Topic	TopicName	The name of a topic.
	TopicDescription	The description of a topic.
	XPathExpression	The XPath 1.0 expression that is used to filter message content.
Namespace	NamespacePrefix	The namespace prefix used for XPath expression.
	NamespaceName	The namespace name used for XPath expression.
Session	SessionID	An identifier generated by the ISBM upon creation of a channel. Identifiers should be made non-obvious and not easily guessable.
	ListenerURI	The URI endpoint that hosts an ISBM Notification Service. Used to indicate when a new message has arrived. See Section 2.7 for more details.
Message	MessageID	An identifier generated by the ISBM upon creation of a message.
	Content	The XML content of a message. No restrictions are placed on the XML.
Publication Message	Expiry	The duration until the expiration of the publication message.

5.2 ISBM Channel Management Services

5.2.1 Create Channel

Name	CreateChannel
Description	Creates a new channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • ChannelType (*ChannelType*) [1] • ChannelDescription (*xs:string*) [0..1] • SecurityToken (*wsse:SecurityTokenReferenceType*) [0..*]
Behavior	• If the combination of ChannelURI and ChannelType are already defined, then no action is taken. • If the ChannelURI already exists but the specified ChannelType does not match the existing ChannelType, then a DuplicateChannelURIFault is thrown. • 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.
Returns	• N/A

Faults	• DuplicateChannelURIFault

5.2.2 Create Topic

Name	CreateTopic
Description	Creates a new topic on a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • TopicName (*xs:string*) [1] • TopicDescription (*xs:string*) [0..1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the TopicName for the channel is already defined, then no action is taken.
Returns	• N/A
Faults	• ChannelFault

5.2.3 Add Security Tokens

Name	AddSecurityTokens
Description	Adds security tokens to a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • SecurityToken (*wsse:SecurityTokenReferenceType*) [1..*] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If a specified SecurityToken is already assigned to the channel, then no action is taken to maintain a distinct list.
Returns	• N/A
Faults	• ChannelFault

5.2.4 Remove Security Tokens

Name	RemoveSecurityTokens
Description	Removes security tokens from a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • SecurityToken (*wsse:SecurityTokenReferenceType*) [1..*] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]

Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If any specified SecurityToken is not assigned to the channel, then an InvalidSecurityTokenFault is thrown. No tokens are removed from the channel, even if they are valid.
Returns	• N/A
Faults	• ChannelFault • InvalidSecurityTokenFault

5.2.5 Delete Channel

Name	DeleteChannel
Description	Deletes a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • The channel, and associated topics and sessions are deleted. No notification is provided to any applications with active sessions.
Returns	• N/A
Faults	• ChannelFault

5.2.6 Delete Topic

Name	DeleteTopic
Description	Deletes a topic from a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • TopicName (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the Topic Name does not exist, then an InvalidTopicFault is thrown. • The topic and associated sessions are deleted. No notification is provided to any applications with active sessions.
Returns	• N/A
Faults	• ChannelFault • InvalidTopicFault

5.2.7 Get Channel

Name	GetChannel
Description	Gets information about a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown.
Returns	• Channel (*Channel*) [1], composed of: o ChannelURI (*xs:string*) [1] o ChannelType (*ChannelType*) [1] o ChannelDescription (*xs:string*) [0..1] o TopicName (*xs:string*) [0..*]
Faults	• ChannelFault

5.2.8 Get Channels

Name	GetChannels
Description	Gets information about all channels.
Input Parameters	• AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..*]
Behavior	• The channels returned are filtered by those that have any of the specified AuthorizationTokens. If a channel does not have tokens assigned, these are returned regardless.
Returns	• Channel (*Channel*) [0..*], composed of: o ChannelURI (*xs:string*) [1] o ChannelType (*ChannelType*) [1] o ChannelDescription (*xs:string*) [0..1] o TopicName (*xs:string*) [0..*]
Faults	• N/A

5.2.9 Get Topics

Name	GetTopics
Description	Gets all topics for a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown.
Returns	• Topic (*Topic*) [0..*], composed of:

	○ TopicName (*xs:string*) [1] ○ TopicDescription (*xs:string*) [0..1]
Faults	• ChannelFault

5.3 ISBM Notification Services

5.3.1 Notify Listener

Name	NotifyListener
Description	Provides a notification of a new message being able to be read for a session. The Listener URI invoked was given when the application desiring notifications subscribed to the channel.
Input Parameters	• SessionID (*xs:string*) [1] • MessageID (*xs:string*) [1] • TopicName (*xs:string*) [0..*] ○ Zero only for consumer read response sessions • RequestMessageID (*xs:string*) [0..1] ○ Allows correlation with original request ○ Only used for consumer read response sessions
Returns	• N/A
Faults	• N/A

5.4 ISBM Provider Publication Services

5.4.1 Open Publication Session

Name	OpenPublicationSession
Description	Opens a publication session for a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the channel type is not a Publication type, then an InvalidOperationFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• ChannelFault • InvalidOperationFault

5.4.2 Post Publication

Name	PostPublication

Description	Posts a publication message on a channel.
Input Parameters	• SessionID (*xs:string*) [1] • MessageContent (*xs:any*) [1] • TopicName (*xs:string*) [1..*] • Expiry (*xs:duration*) [0..1]
Behavior	• The ISBM Service Provider creates a message with the MessageContent and a MessageID that uniquely identifies message, and makes it available for subscribers. • If the SessionID does not exist then an InvalidSessionFault is thrown. • If any of the TopicNames do not exist for the channel (where the channel is implied from the session) or do not belong to the channel, then an InvalidTopicFault is thrown. • A negative Expiry duration is considered equivalent to a zero duration.
Returns	• MessageID (*xs:string*) [1]
Faults	• InvalidSessionFault • InvalidTopicFault

5.4.3 Expire Publication

Name	ExpirePublication
Description	Expires a posted publication.
Input Parameters	• SessionID (*xs:string*) [1] • MessageID (*xs:string*) [1]
Behavior	• If the SessionID does not exist then an InvalidSessionFault is thrown. • If the MessageID does not correspond with the SessionID or the corresponding message has already expired, then no action is taken.
Returns	• N/A
Faults	• InvalidSessionFault

5.4.4 Close Publication Session

Name	ClosePublicationSession
Description	Closes a publication session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist (non-existent or already closed) then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault

5.5 ISBM Consumer Publication Services

5.5.1 Open Subscription Session

Name	OpenSubscriptionSession
Description	Opens a subscription session for a channel.

Input Parameters	• ChannelURI (*xs:string*) [1]
	• TopicName (*xs:string*) [1..*]
	• ListenerURI (*xs:string*) [0..1]
	• XPathExpression (*xs:string*) [0..1]
	• XPathNamespace (*Namespace*) [0..*], composed of:
	  o NamespacePrefix (*xs:string*) [1]
	  o NamespaceName (*xs:string*) [1]
	• AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]

Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the channel type is not a Publication type, then an InvalidOperationFault is thrown. • If any of the TopicNames do not exist for the channel or do not belong to the channel, then an InvalidTopicFault is thrown. • If multiple NamespacePrefixes exist with different NamespaceNames, then a DuplicateNamespacePrefixFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• ChannelFault • InvalidOperationFault • InvalidTopicFault • DuplicateNamespacePrefixFault

5.5.2 Read Publication

Name	ReadPublication
Description	Returns the first non-expired publication message (if any) after the specified last message (even if the specified last message has expired) for the session.
Input Parameters	• SessionID (*xs:string*) [1] • LastMessageID (*xs:string*) [0..1]
Behavior	• If the SessionID does not exist then an InvalidSessionFault is thrown.

Returns	• PublicationMessage (*PublicationMessage*) [0..1], composed of:
	  o MessageID (*xs:string*) [1]
	  o MessageContent (*xs:any*) [1]

	○ TopicName (*xs:string*) [1..*]
Faults	• InvalidSessionFault

5.5.3 Close Subscription Session

Name	CloseSubscriptionSession
Description	Closes a subscription session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist (non-existent or already closed) then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault

5.6 ISBM Provider Request Services

5.6.1 Open Read Request Session

Name	OpenReadRequestSession
Description	Opens a read request session for a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • TopicName (*xs:string*) [1..*] • ListenerURI (*xs:string*) [0..1] • XPathExpression (*xs:string*) [0..1] • XPathNamespace (*Namespace*) [0..*], composed of: ○ NamespacePrefix (*xs:string*) [1] ○ NamespaceName (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the channel type is not a Request type, then an InvalidOperationFault is thrown. • If any of the Topic Names do not exist for the channel or do not belong to the channel, then an InvalidTopicFault is thrown. • If multiple Namespace Prefixes exist with different NamespaceNames, then a DuplicateNamespacePrefixFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• ChannelFault • InvalidOperationFault • InvalidTopicFault

	• DuplicateNamespacePrefixFault

5.6.2 Read Request

Name	ReadRequest
Description	Returns the first request message in the message queue for the session. Note: this service does not remove the message from the message queue.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist then an InvalidSessionFault is thrown.
Returns	• RequestMessage (*RequestMessage*) [0..1], composed of: o MessageID (*xs:string*) [1] o MessageContent (*xs:any*) [1] o TopicName (*xs:string*) [1..*]
Faults	• InvalidSessionFault

5.6.3 Remove Request

Name	RemoveRequest
Description	Deletes the first request message in the message queue for the session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault

5.6.4 Close Read Request Session

Name	CloseReadRequestSession
Description	Closes a read request session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist (non-existent or already closed) then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault

5.6.5 Open Post Response Session

Name	OpenPostResponseSession
Description	Opens a post response session for a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]

Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the channel type is not a Response type, then an InvalidOperationFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• ChannelFault • InvalidOperationFault

5.6.6 Post Response

Name	PostResponse
Description	Posts a response message on a channel.
Input Parameters	• SessionID (*xs:string*) [1] • RequestMessageID (*xs:string*) [1] • MessageContent (*xs:any*) [1]
Behavior	• The ISBM Service Provider creates a message with the MessageContent and a MessageID that uniquely identifies message, and makes the response available to the requesting application. • If the SessionID does not exist then an InvalidSessionFault is thrown. • The ISBM Service Provider is not required to validate that the RequestMessageID is valid.
Returns	• MessageID (*xs:string*) [1]
Faults	• InvalidSessionFault

5.6.7 Close Post Response Session

Name	ClosePostResponseSession
Description	Closes a post response session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist (non-existent or already closed) then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault

5.7 ISBM Consumer Request Services

5.7.1 Open Post Request Session

Name	OpenPostRequestSession
Description	Opens a post request session for a channel.

Input Parameters	• ChannelURI (*xs:string*) [1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the channel type is not a Request type, then an InvalidOperationFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• ChannelFault • InvalidOperationFault

5.7.2 Post Request

Name	PostRequest
Description	Posts a request message on a channel.
Input Parameters	• SessionID (*xs:string*) [1] • MessageContent (*xs:any*) [1] • TopicName (*xs:string*) [1]
Behavior	• The ISBM Service Provider creates a message with the MessageContent and a MessageID that uniquely identifies message, and makes the message available to the appropriate provider applications on the channel. • If the SessionID does not exist then an InvalidSessionFault is thrown. • If any of the TopicNames do not exist for the channel (where the channel is implied from the session) or do not belong to the channel, then an InvalidTopicFault is thrown.
Returns	• MessageID (*xs:string*) [1]
Faults	• InvalidSessionFault • InvalidTopicFault

5.7.3 Close Post Request Session

Name	ClosePostRequestSession
Description	Closes a post request session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist (non-existent or already closed) then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault

5.7.4 Open Read Response Session

Name	OpenReadResponseSession
Description	Opens a read response session for a channel.
Input Parameters	• ChannelURI (*xs:string*) [1] • ListenerURI (*xs:string*) [0..1] • AuthorizationToken (*wsse:SecurityTokenReferenceType*) [0..1]
Behavior	• If the ChannelURI does not exist, or if the specified channel is assigned security tokens and the specified AuthorizationToken does not match a token assigned to the specified channel, then a ChannelFault is thrown. • If the channel type is not a Response type, then an InvalidOperationFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• ChannelFault • InvalidOperationFault

5.7.5 Read Response

Name	ReadResponse
Description	Returns the first response message associated with the request. Note: this service does not remove the message from the message queue.
Input Parameters	• SessionID (*xs:string*) [1] • RequestMessageID (*xs:string*) [1]
Behavior	• If the SessionID does not exist then an InvalidSessionFault is thrown.

Returns	• ResponseMessage (*ResponseMessage*) [0..1], composed of:
	  ○ MessageID (*xs:string*) [1]
	  ○ MessageContent (*xs:any*) [1]

Faults	• InvalidSessionFault

5.7.6 Remove Response

Name	RemoveResponse
Description	Deletes the first request message in the message queue for the session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault

5.7.7 Close Read Response Session

Name	CloseReadResponseSession

Description	Closes a read response session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist (non-existent or already closed) then an InvalidSessionFault is thrown.
Returns	• N/A
Faults	• InvalidSessionFault