OpenO&M™
Information Service Bus Model (ISBM) Specification
v1.0 RC5

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.

OpenO&M Information Service Bus Model (ISBM)
Copyright 2011. MIMOSA
All Rights Reserved. http://www.mimosa.org

Parts derived from WBF B2MML-V0401
"The Business To Manufacturing Markup Language (B2MML) is used courtesy of WBF."

Copyright © MIMOSA 2011. All Rights Reserved.

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.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to MIMOSA, except as needed for the purpose of developing any document or deliverable produced by a MIMOSA Technical Committee (in which case the rules applicable to copyrights, as set forth in the MIMOSA IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by MIMOSA or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and MIMOSA DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

MIMOSA requests that any MIMOSA Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this MIMOSA Final Deliverable, to notify MIMOSA TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the MIMOSA Technical Committee that produced this deliverable.

MIMOSA invites any party to contact the MIMOSA TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this MIMOSA Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the MIMOSA Technical Committee that produced this MIMOSA Final Deliverable. MIMOSA may include such claims on its website, but disclaims any obligation to do so.

MIMOSA takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this MIMOSA Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on MIMOSA' procedures with respect to rights in any document or deliverable produced by a MIMOSA Technical Committee can be found on the MIMOSA website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this MIMOSA Final Deliverable, can be obtained from the MIMOSA TC Administrator. MIMOSA makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

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 6
2 OpenO&M Information Service Bus Model 7
2.1 Interface Model 7
2.2 Application to Application Data Exchange 8
2.3 Transaction Model 9
2.4 Communicating Applications 9
2.5 Managed Communication Channels 10
2.6 ISBM Channel Management Services 11
2.7 ISBM Notification Services 12
2.8 ISBM Publish-Subscribe Services 12
2.9 ISBM Request and Response Channel Services 15
3 ISBM Technical Requirements 19
3.1 Channel Names 19
3.1.1 ISBM Root 19
3.1.2 Channel Scope 19
3.1.3 Information Scope 20
3.1.4 Channel Use 20
3.2 Topics 20
3.2.1 XPath Filtering 20
3.3 Session Queuing 21
3.4 Security 21
4 ISBM Service Provider Considerations 22
4.1 Security Considerations 22
4.2 Notification 22
4.3 Data Format Validation 22
4.4 Allowed Application Checking 22
4.5 Data Exchange Logging 22
4.6 Common Error Handling 22
4.7 Data Transformation Services 23
4.8 Cross Company Bridge 23
5 Service Definitions 25
5.1 Data Model 25
5.1.1 Data Dictionary 25
5.2 ISBM Channel Management Services 26
5.2.1 Create Channel 26
5.2.2 Create Topic 27
5.2.3 Delete Channel 27
5.2.4 Delete Topic 27
5.2.5 Get Channels 28
5.2.6 Get Session 28

5.2.7 Get Sessions 29
5.2.8 Get Topic 29
5.2.9 Get Topics 30
5.3 ISBM Notification Services 30
5.3.1 Notify Listener 30
5.4 ISBM Provider Publication Services 30
5.4.1 Open Publication Session 30
5.4.2 Post Publication 31
5.4.3 Close Publication Session 31
5.5 ISBM Consumer Publication Services 32
5.5.1 Open Subscription Session 32
5.5.2 Read Publication 32
5.5.3 Remove Publication 32
5.5.4 Close Subscription Session 33
5.6 ISBM Provider Request Services 33
5.6.1 Open Read Request Session 33
5.6.2 Read Request 33
5.6.3 Remove Request 34
5.6.4 Close Read Request Session 34
5.6.5 Open Post Response Session 34
5.6.6 Post Response 35
5.6.7 Close Post Response Session 35
5.7 ISBM Consumer Request Services 35
5.7.1 Open Post Request Session 35
5.7.2 Post Request 36
5.7.3 Close Post Request Session 36
5.7.4 Open Read Response Session 36
5.7.5 Read Response 37
5.7.6 Remove Response 37
5.7.7 Close Read Response Session 37

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[1] or to any other message or file exchange system that offers guaranteed message and storage or caching of exchanged messages.

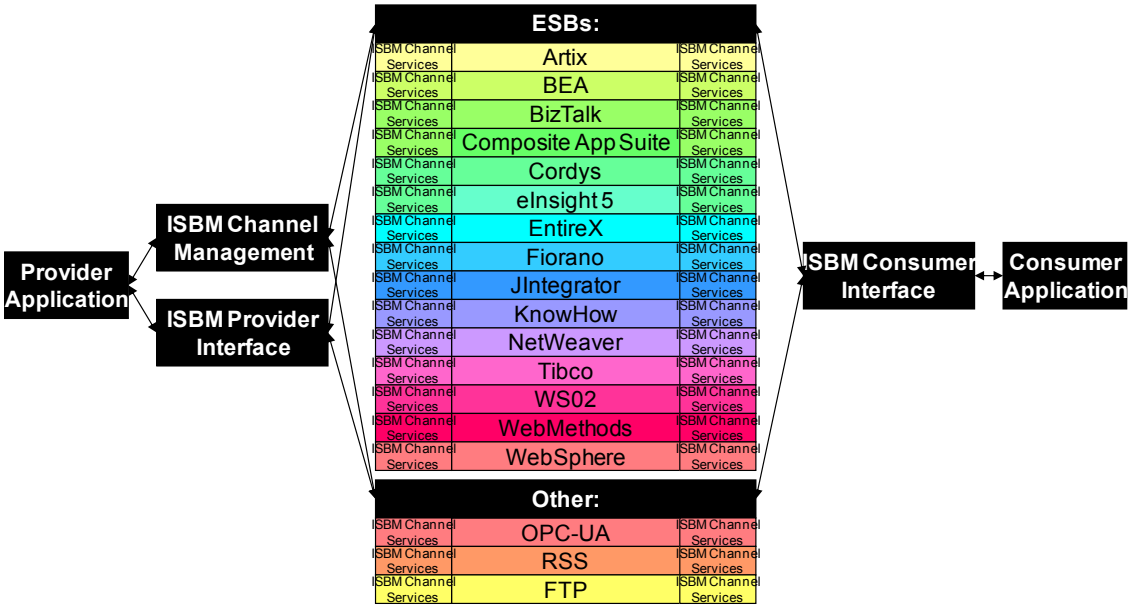


Figure 1 - ISBM Interface to ESB and Other Service Providers

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.

[1] See Annex B for a brief discussion on ESBs.

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 CIR, 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.

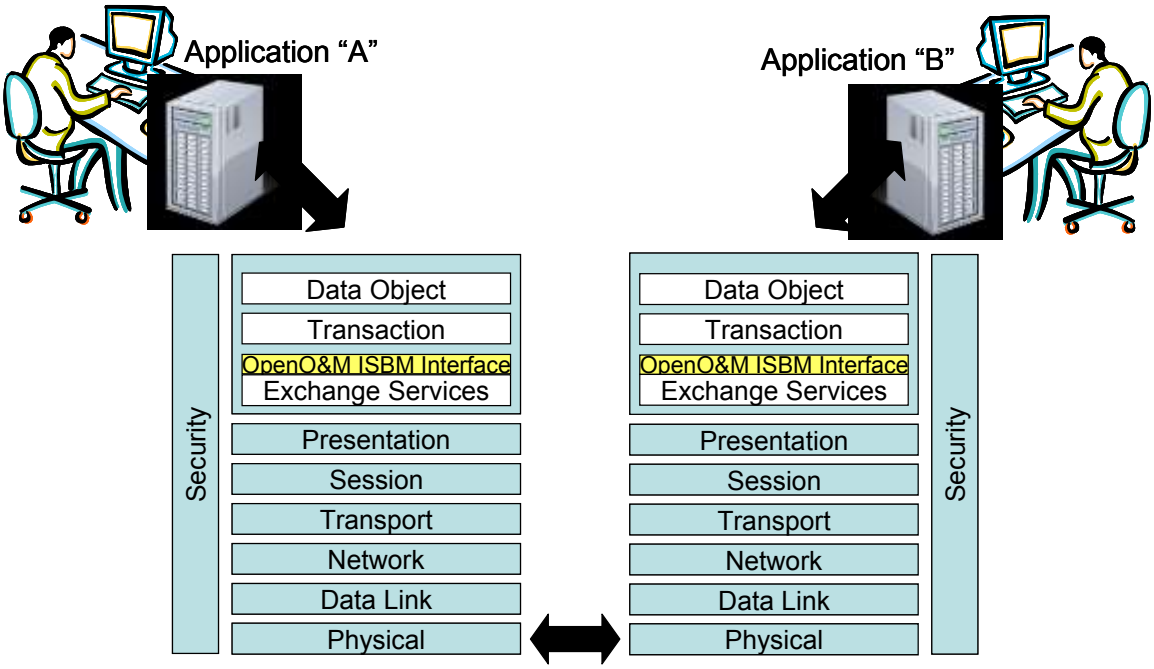


Figure 2 – Loosely Coupled Application Communication Stack

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, OpenO&M CIR 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.

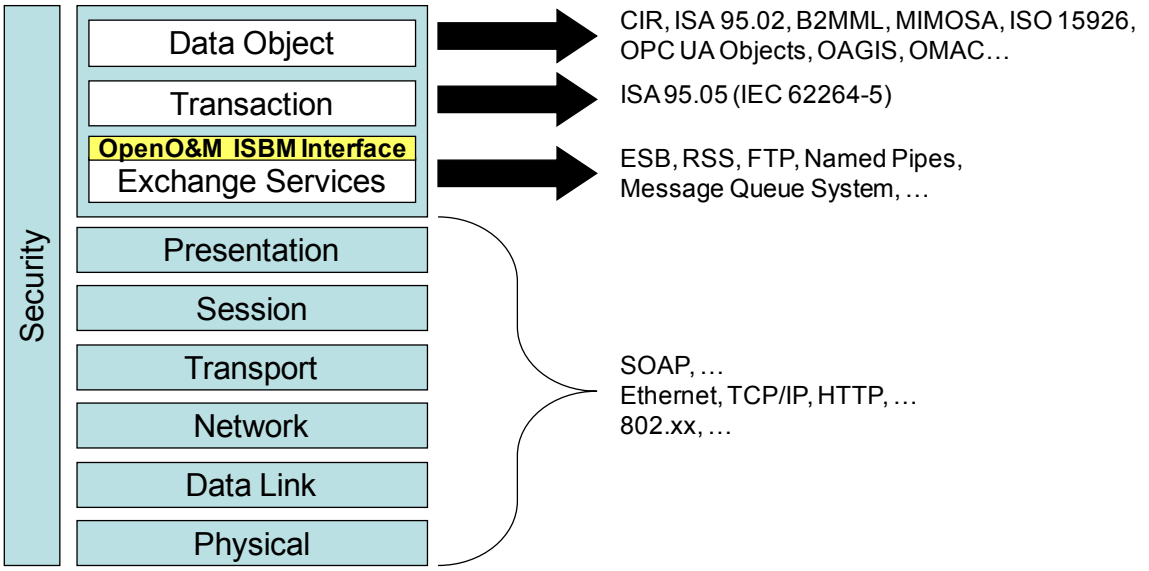


Figure 3 – Defined Standards at Each Application Sublevel

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[2].

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 with multiple subscribers and multiple publishers, where the publishers and subscribers have not direct knowledge of other applications.
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.

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).

[2] 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.

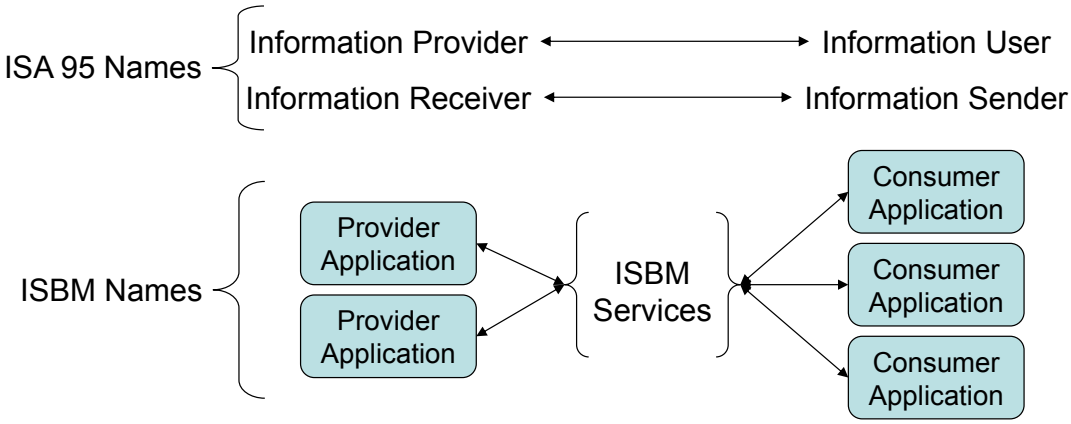


Figure 4 – OpenO&M Information Service Bus Model Names

While the ISA 95.05 verbs (e.g. GET, SYNC and CANCEL) can be used with both transaction model types (i.e. publish-subscribe and request-response), the semantics of the GET verb only aligns with the request-response model.

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 an ISBM *Channel*.
- The services provided for each ISBM Channel are the *ISBM Channel Services*.
- An ISBM *Channel Name* is identified using a URI. A 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 Services*.
- A recommended structure for the *ISBM Channel* hierarchy is defined in this document.

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

1. An ISBM *Channel* may be created to support either publication services, request services, or response services.
2. A *Provider Application* may post publications to an ISBM *Publication Channel*.
3. *Consumer Applications* may subscribe to publication notifications (if supported by the specific *ISBM Publication Channel Service*) and may read publications. If notifications are not supported, then the *Consumer Application* may poll the *ISBM Publication Channel* using the read publication service.
4. A *Consumer Application* may post requests to an ISBM *Request Channel*.
5. A *Provider Application* may subscribe to request notifications (if supported by the specific *ISBM Request Channel Service*) and may read requests. If notifications are not supported, then the *Provider Application* may poll the *ISBM Request Channel* using the read request service.
6. *ISBM 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.

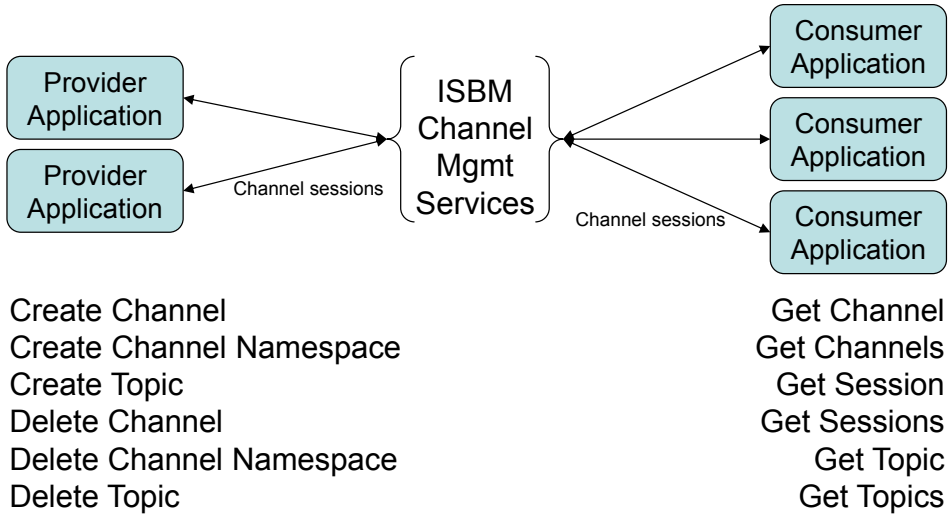


Figure 5 – ISBM Channel Management Services

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 Notify Listener service allows the ISBM 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. See Section 3.3 for details on message queues and notification.

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 ppplications may subscribe to callback notifications of a new message in their session queue or can poll to read publications. Topics provide a subscription filtering mechanism.

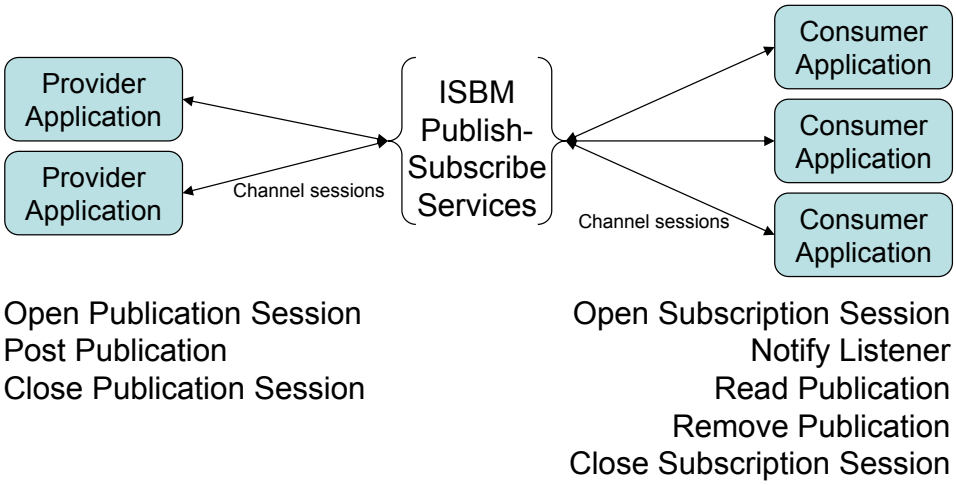


Figure 6 – ISBM Publish-Subscribe Services

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[3]. When the provider application has determined that data should be published, it posts publications with a message topic.

A consumer application subscribes to the ISBM publication channel using a channel ID and a list of topics. When a new message with the right topic is posted, the consumer application is notified of the posting and then reads the new publication message from the ISBM channel. When the consumer application no longer needs data, or is exiting, it unsubscribes from the ISBM channel.

[3] It is assumed that the appropriate channels, namespaces and topics have been created prior to the scenario

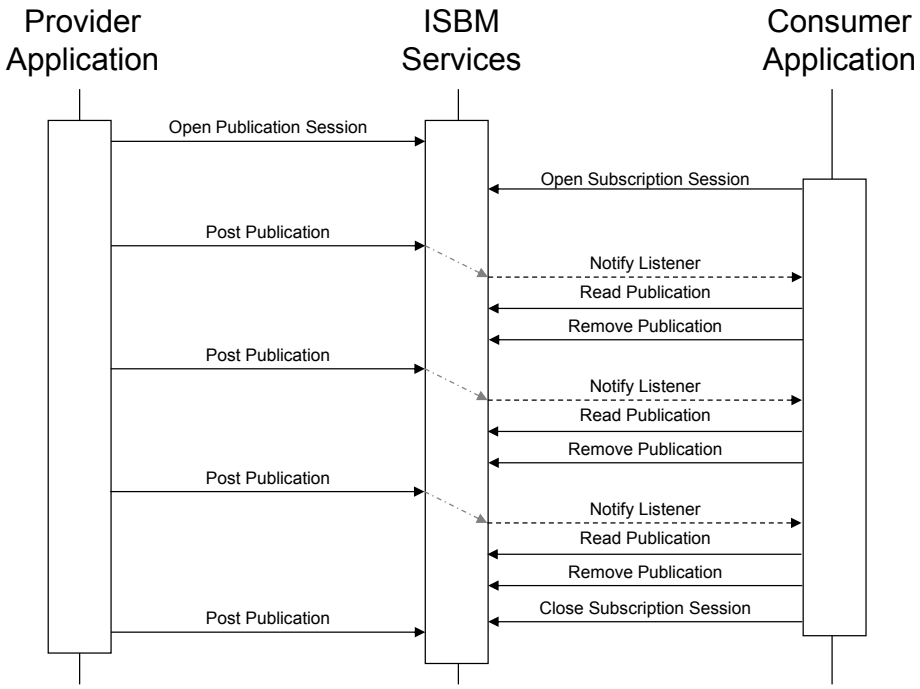


Figure 7 – Publish-subscribe scenario with notification

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 would poll the ISBM channel for publications either periodically or based on some local event. The returned information from the Read Publication indicates if a new publication was returned.

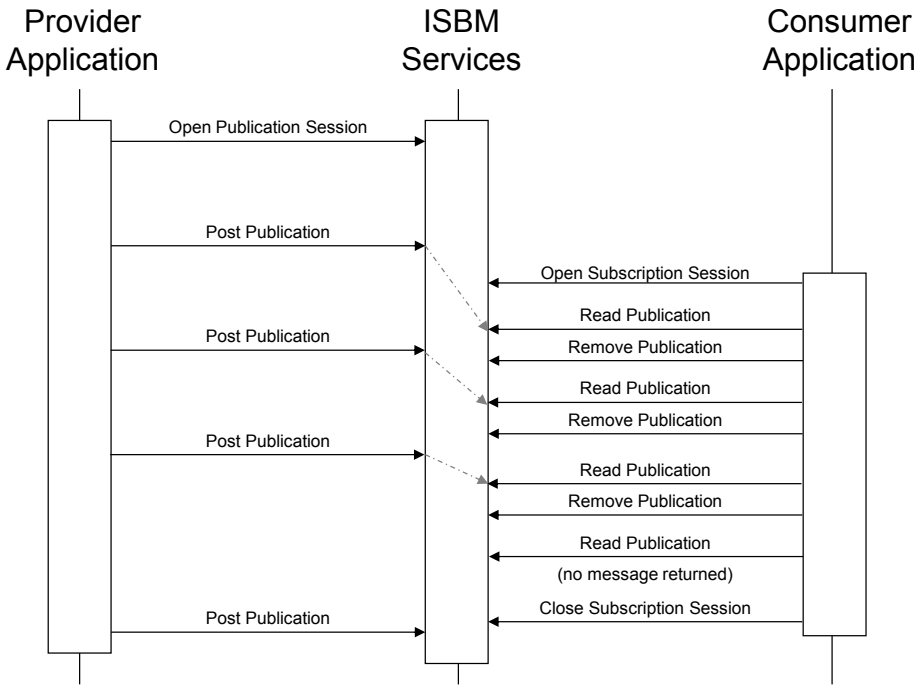


Figure 8 – Publish-subscribe scenario without notification

More than one provider application may use the same publication channel. The scenario shown in Figure 9 has two provider applications. For example, one application could publish changes for material definitions while another may publish changes for material lots.

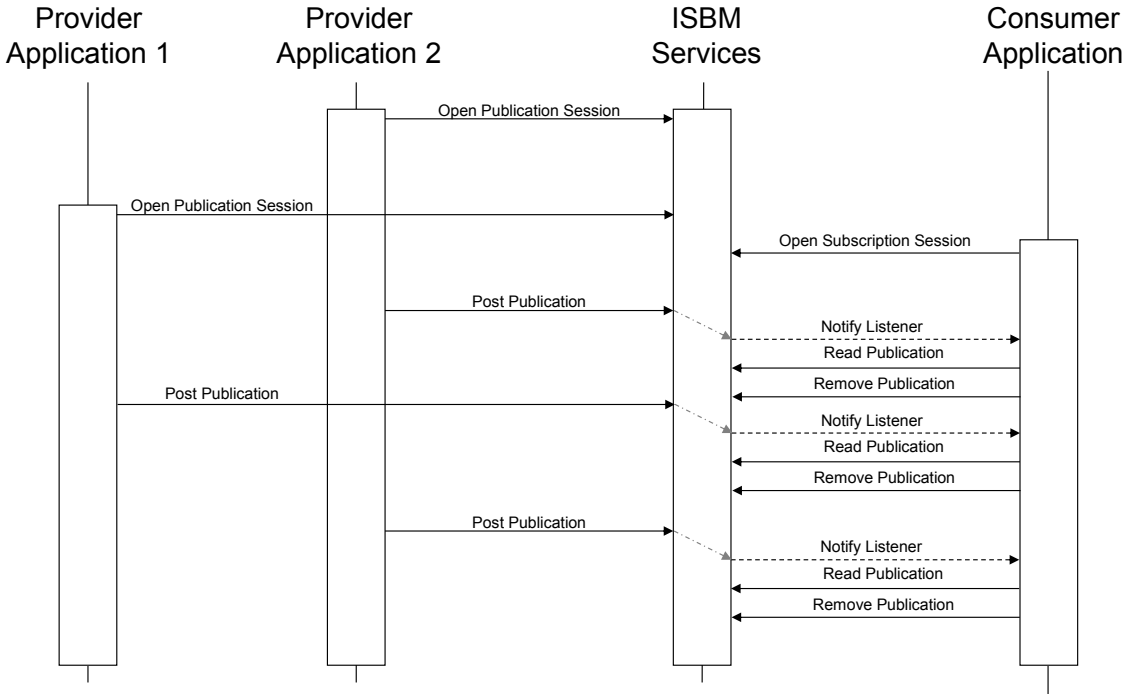
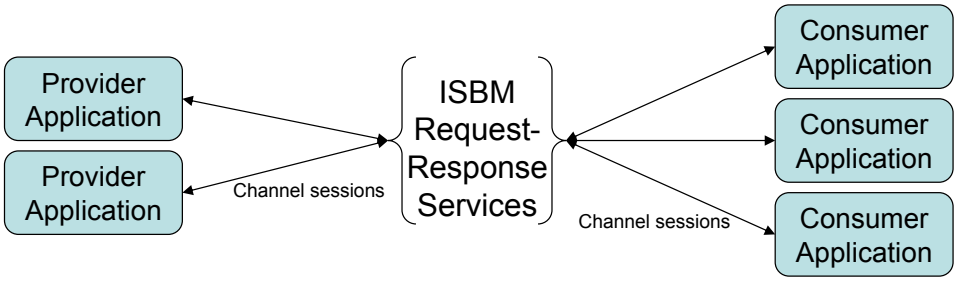


Figure 9 – Publish-subscribe scenario with multiple provider applications

2.9 ISBM Request and Response Channel Services

The ISBM Request-Response Channel Services are shown in Figure 10. 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	Open Post Request Session
Notify Listener	Post Request
Read Request	Close Post Request Session
Remove Request	Open Read Response Session
Close Read Request Session	Notify Listener
Open Post Response Session	Read Response
Post Response	Remove Response
Close Post Response Session	Close Read Response Session

Figure 10 – Services for Request/Response

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 11.

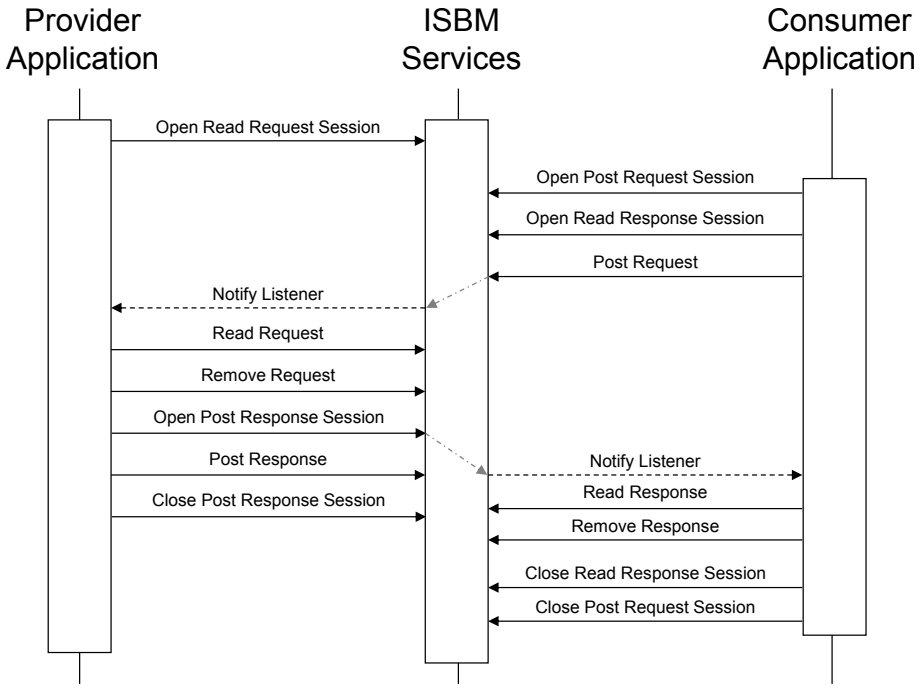


Figure 11 – Request-Response scenario with notification

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.

If the applications or ISBM services do not support notification, then the provider and consumer applications may poll for a request or response. Figure 12 illustrates a request-response scenario where the consumer application must poll for a response.

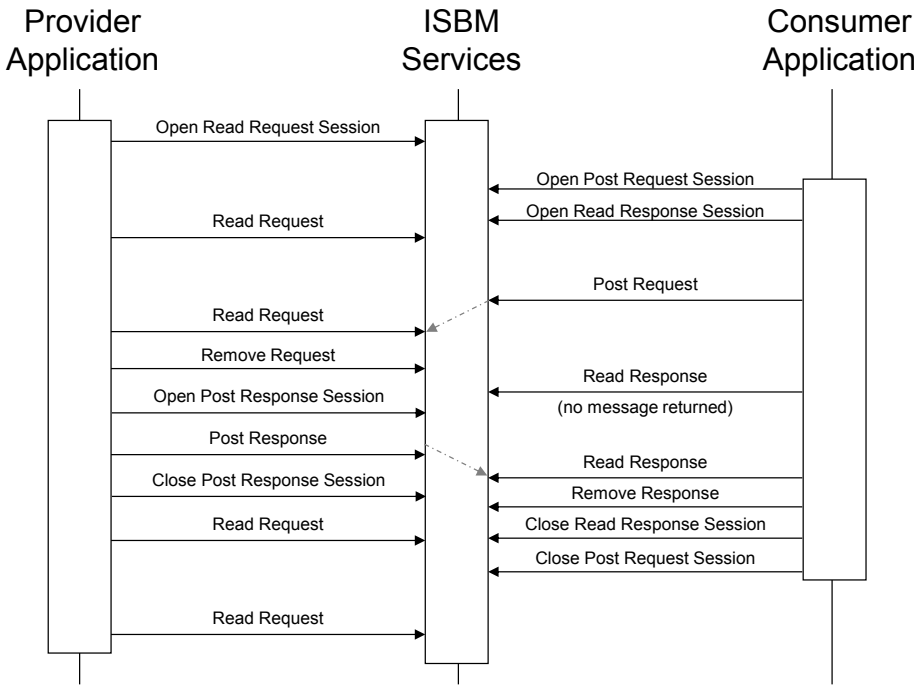


Figure 12 – Request-Response scenario without notification

Figure 13 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.

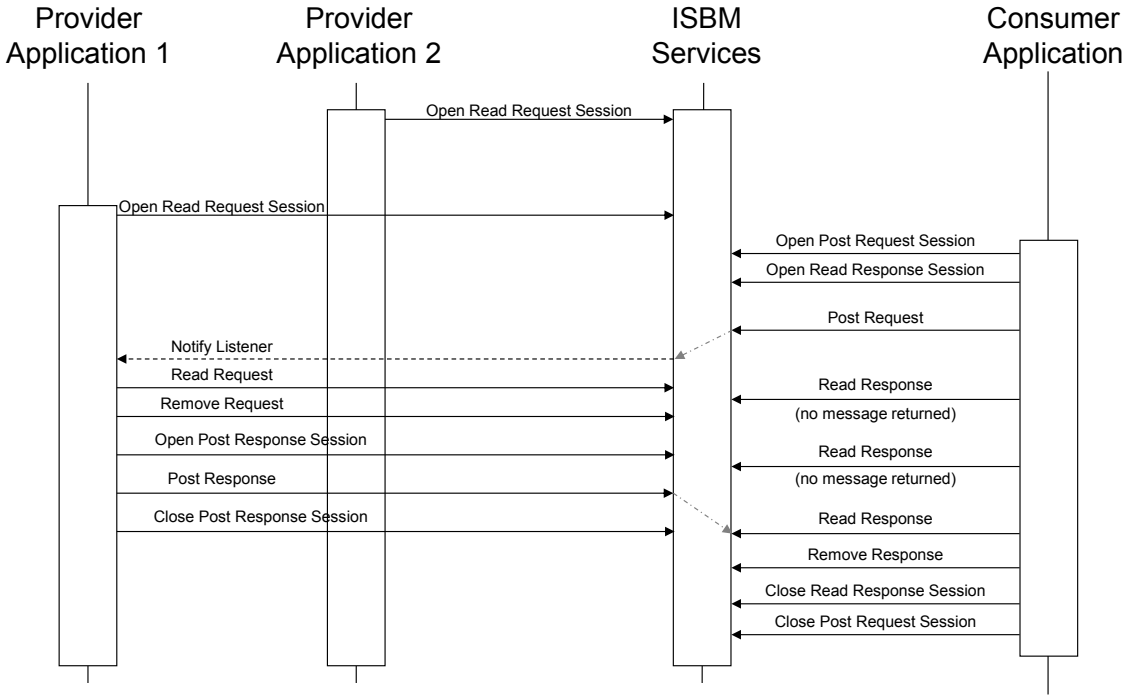


Figure 13 – Request-response scenario with multiple provider applications

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 Names

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

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

For example:

\AJAXEnterprises\Company\Material\Checkpoint
\AJAXEnterprises\Company\Material\Request
\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 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.

3.2.1 XPath Filtering

To allow content filtering of messages, an XPath expression can be optionally added to a topic to provide a filtering mechanism. The XPath expression must be defined as an XPath v1.0 expression that returns a node set that is considered valid XML. As valid XML has a single root element, an XPath evaluation that returns multiple nodes is considered invalid and a fault is returned to the provider application. For an XPath expression that use namespaces, multiple namespace prefixes and names can be added upon topic creation.

Before a message is placed in a consumer session queue, any corresponding topic XPath expressions should first be evaluated on the message content. The ISBM will replace the message content with the results of the XPath evaluation.

3.3 Session Queuing

The ISBM must maintain a queue for each session for a consumer application (i.e. subscriber sessions, request sessions, and response sessions). The queue stores messages routed to the consuming application, from which the application will read and delete the top most message. The message order in the queue is determined by when a provider application *invokes* the call to put the message on the ISBM (and not by the completion of the call). In a notification scenario, the ISBM must delay notifying any consumer applications of the pending message until notifications of earlier invoked messages have been sent.

In the below example, Provider Application 2 starts the PostPublication call before and Provider Application 1 starts its PostPublication call. While Provider Application 1 completes the call first (due to a smaller message size), the Consumer Application will not receive a notification (or in a polling scenario, will not be able to read the message) until Provider Application 2's PostPublication call is complete. At this point, notifications are sent consecutively (in order of invocation) to the Consumer Application.

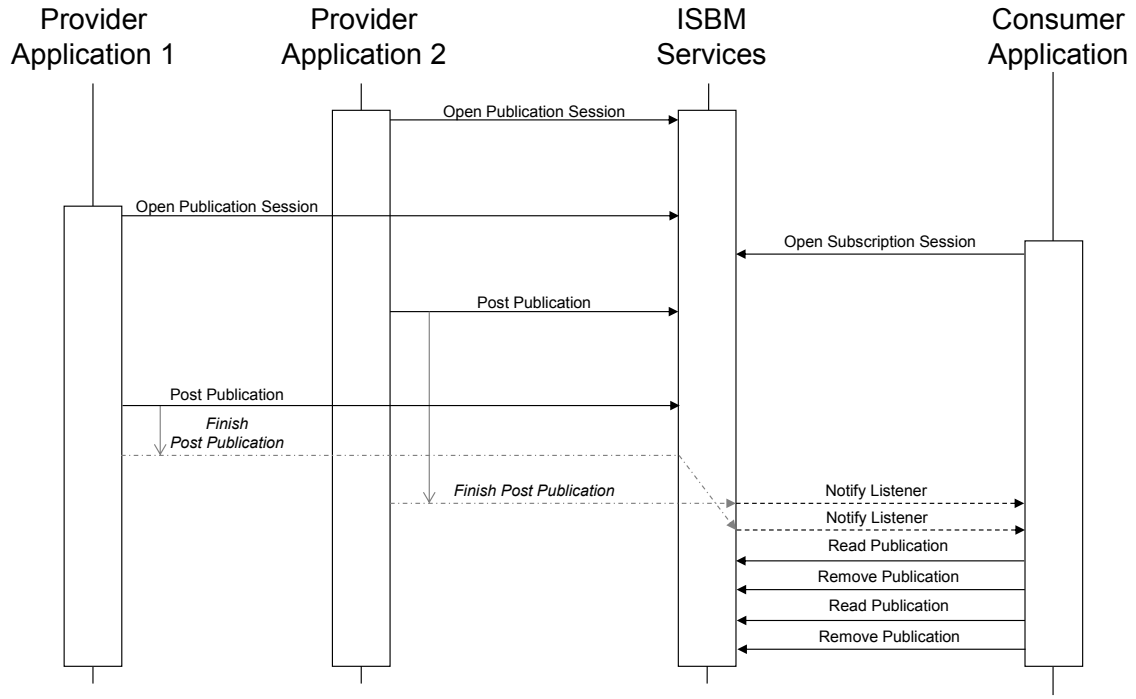


Figure 14 – Message Queuing and Delayed Notification

All messages in a session queue can be safely discarded once the consumer application has closed the session (i.e. unsubscribe publication, request or response channels).

3.4 Security

A security model is not specified in this version of the ISBM and will be addressed thoroughly in an upcoming version. In absence of a security model, it is advised that Transport Layer Security be used if authentication, encryption or signatures are required between applications and the ISBM.

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. 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.
4. 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*.

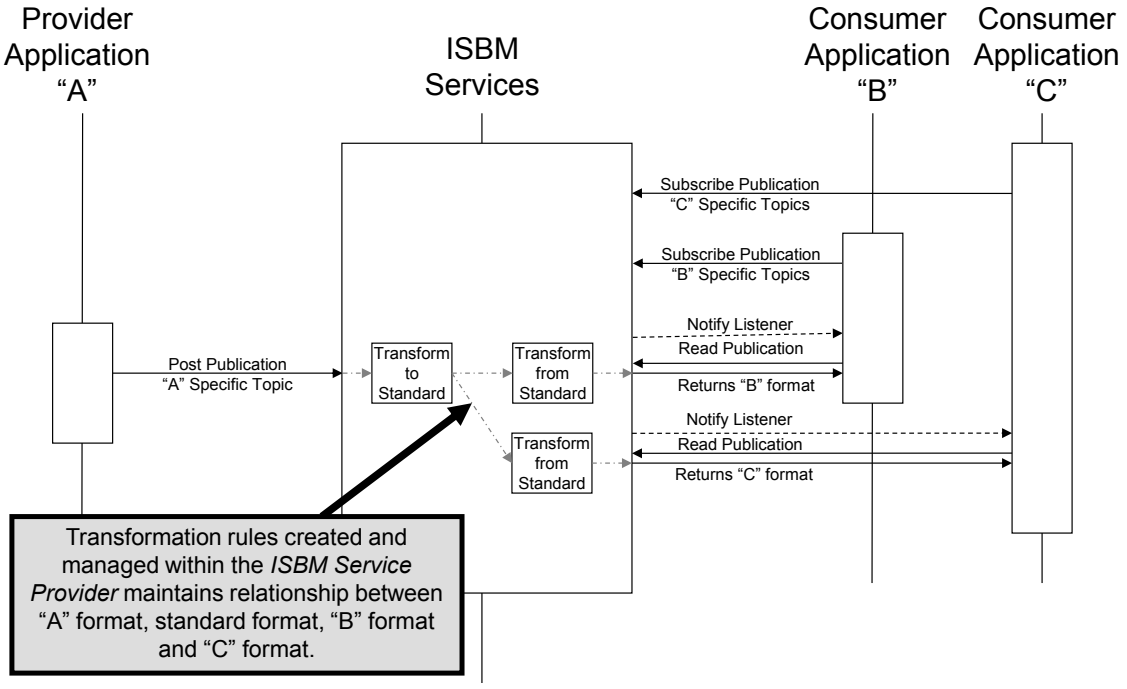


Figure 15 – Transformation Services with the ISBM Service Provider

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 16. 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.

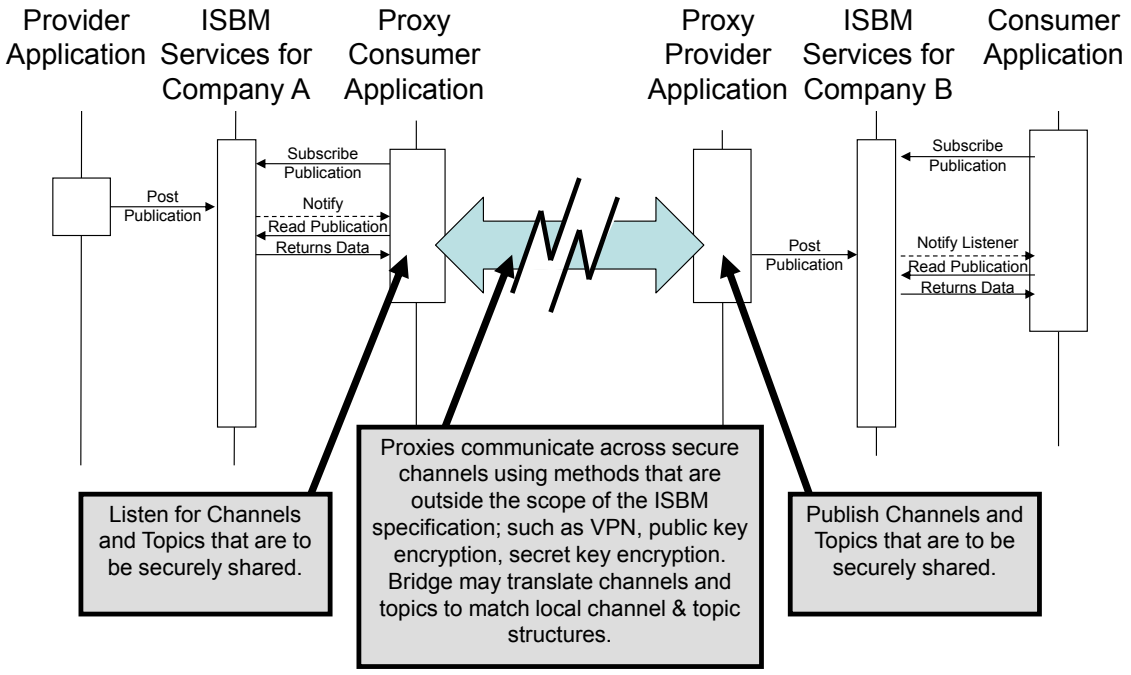


Figure 16 – Cross Company Bridge between multiple ISBMs

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 used by the ISBM services and their relationships, a data model is presented below. It is not prescribed as an implementation data model but can serve as the basis for one.

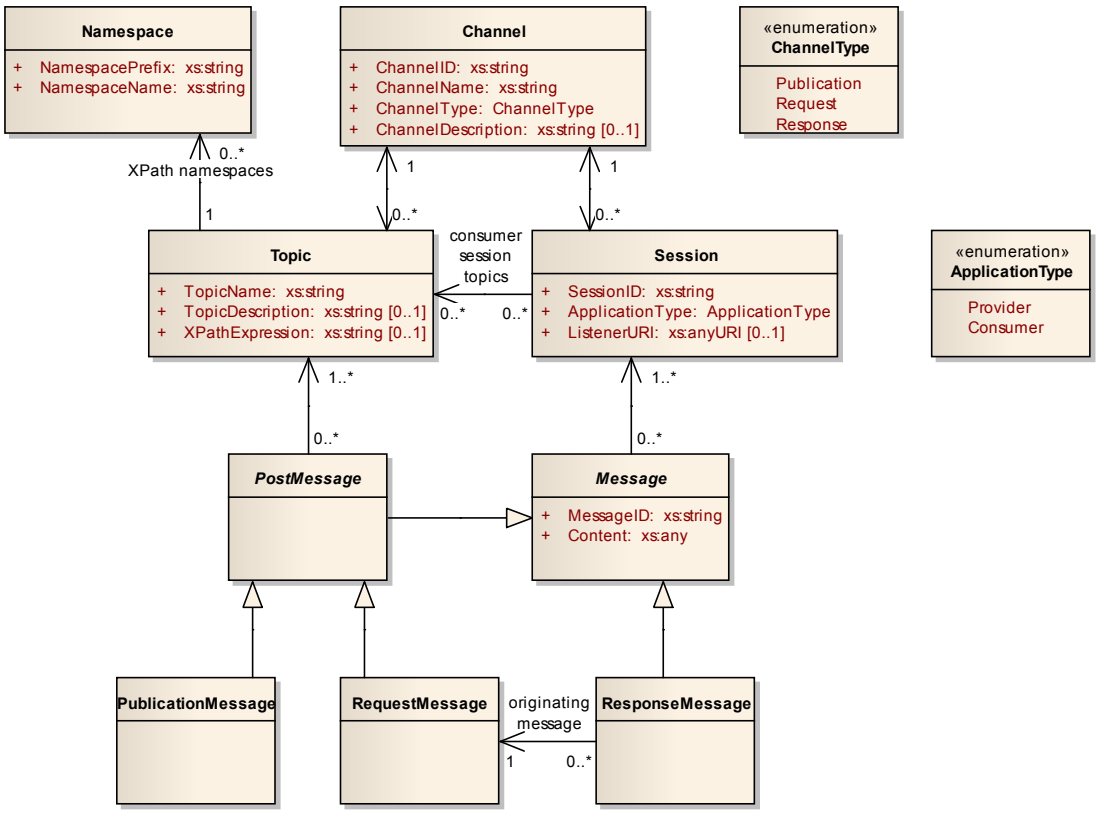


Figure 17 – Data Model

5.1.1 Data Dictionary

Entity	Attribute	Description
Channel	ChannelID	An identifier generated by the ISBM upon creation of a channel. Identifiers can be made non-obvious and not easily guessable to provide security through obscurity.
	ChannelName	The name of a channel. See Section 3.1 for details on the name 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.
	ChannelDescription	The description of a channel.

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 can be made non-obvious and not easily guessable to provide security through obscurity.
	ApplicationType	Indicates whether the session is for a provider application or a consumer application. This data element is inferred when a particular session creation service is invoked.
	ListenerURI	The URI endpoint that hosts an ISBM Notification Service. Used to indicate when a message has been queued for a consumer application. 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.
Response Message	RequestMessageID	The originating request message ID for a response message in a request-response scenario. This is set by the provider application to allow a consumer application to correlate for which request is this response for.

5.2 ISBM Channel Management Services

5.2.1 Create Channel

Name	CreateChannel
Description	Creates a new ISBM channel.
Input Parameters	• ChannelName (*xs:string*) [1] • ChannelType (*ChannelType*) [1] • ChannelDescription (*xs:string*) [0..1]
Behavior	• The ISBM creates a channel and returns the ChannelID that uniquely identifies the channel. • If the combination of ChannelName and ChannelType are already defined, then a DuplicateChannelFault is thrown.
Returns	• ChannelID (*xs:string*) [1]
Faults	• DuplicateChannelFault

5.2.2 Create Topic

Name	CreateTopic
Description	Creates a new topic on a channel.
Input Parameters	• ChannelID (*xs:string*) [1] • TopicName (*xs:string*) [1] • TopicDescription (*xs:string*) [0..1] • XPathExpression (*xs:string*) [0..1] • XPathNamespace (*Namespace*) [0..*]
Behavior	• If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the TopicName for the channel is already defined, then a DuplicateTopicFault is thrown. • If multiple Namespace Prefixes exist with different NamespaceNames, then a DuplicateNamespacePrefixFault is thrown.
Returns	• N/A
Faults	• InvalidChannelFault • DuplicateTopicFault • DuplicateNamespacePrefixFault

5.2.3 Delete Channel

Name	DeleteChannel
Description	Deletes an ISBM channel.
Input Parameters	• ChannelID (*xs:string*) [1]
Behavior	• The channel and associated topics, sessions and queues are deleted. No notification is provided to any applications with active sessions. • If the ChannelID does not exist, then an InvalidChannelFault is thrown.
Returns	• N/A
Faults	• InvalidChannelFault

5.2.4 Delete Topic

Name	DeleteTopic
Description	Deletes a topic from a channel.
Input Parameters	• ChannelID (*xs:string*) [1] • TopicName (*xs:string*) [1]
Behavior	• The topic and associated sessions and queues are deleted. No notification is provided to any applications with active sessions. • If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the Topic Name does not exist, then an InvalidTopicFault is thrown.

Returns	• N/A
Faults	• InvalidChannelFault • InvalidTopicFault

5.2.5 Get Channel

Name	GetChannel
Description	Gets information about a channel.
Input Parameters	• ChannelName (*xs:string*) [1] • ChannelType (*ChannelType*) [1]
Behavior	• If the channel does not exist, then an InvalidChannelFault is thrown.

Returns	• Channel (*Channel*) [1], composed of:
	  ○ ChannelID (*xs:string*) [1]
	  ○ ChannelName (*xs:string*) [1]
	  ○ ChannelType (*ChannelType*) [1]
	  ○ ChannelDescription (*xs:string*) [0..1]
	  ○ TopicName (*xs:string*) [0..*]
	  ○ SessionID (*xs:string*) [0..*]

Faults	• InvalidChannelFault

5.2.6 Get Channels

Name	GetChannels
Description	Gets all ISBM channels.
Input Parameters	• N/A

Returns	• Channel (*Channel*) [0..*], composed of:
	  ○ ChannelID (*xs:string*) [1]
	  ○ ChannelName (*xs:string*) [1]
	  ○ ChannelType (*ChannelType*) [1]
	  ○ ChannelDescription (*xs:string*) [0..1]
	  ○ TopicName (*xs:string*) [0..*]
	  ○ SessionID (*xs:string*) [0..*]

5.2.7 Get Session

Name	GetSession
Description	Gets information about a session.
Input Parameters	• SessionID (*xs:string*) [1]
Behavior	• If the SessionID does not exist, then an InvalidSessionFault is thrown.
Returns	• Session (*Session*) [1], composed of:

	○ SessionID (*xs:string*) [1] ○ ChannelID (*xs:string*) [1] ○ ApplicationType (*ApplicationType*) [1] ○ TopicName (*xs:string*) [0..*] ▪ Only for consumer sessions ○ ListenerURI (xs:anyURI) [0..1] ▪ Only for consumer sessions
Faults	• InvalidSessionFault

5.2.8 Get Sessions

Name	GetSessions
Description	Gets all sessions for a channel.
Input Parameters	• ChannelID (*xs:string*) [1]
Behavior	• If the ChannelID does not exist, then an InvalidChannelFault is thrown.
Returns	• Session (*Session*) [0..*], composed of: ○ SessionID (*xs:string*) [1] ○ ChannelID (*xs:string*) [1] ○ ApplicationType (*ApplicationType*) [1] ○ TopicName (*xs:string*) [0..*] ▪ Only for consumer sessions ○ ListenerURI (xs:anyURI) [0..1] ▪ Only for consumer sessions
Faults	• InvalidChannelFault

5.2.9 Get Topic

Name	GetTopic
Description	Gets information about a topic.
Input Parameters	• ChannelID (*xs:string*) [1] • TopicName (*xs:string*) [1]
Behavior	• If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the TopicName does not exist, then an InvalidTopicFault is thrown.
Returns	• Topic (*Topic*) [1], composed of: ○ TopicName (*xs:string*) [1] ○ ChannelID (*xs:string*) [1] ○ TopicDescription (*xs:string*) [0..1] ○ XPathExpression (*xs:string*) [0..1] ○ XPathNamespace (*Namespace*) [0..*]

Faults	• InvalidChannelFault • InvalidTopicFault

5.2.10 Get Topics

Name	GetTopics
Description	Gets all topics for a channel.
Input Parameters	• ChannelID (*xs:string*) [1]
Behavior	• If the ChannelID does not exist, then an InvalidChannelFault is thrown.

Returns	• Topic (*Topic*) [0..*], composed of:
	  o TopicName (*xs:string*) [1]
	  o ChannelID (*xs:string*) [1]
	  o TopicDescription (*xs:string*) [0..1]
	  o XPathExpression (*xs:string*) [0..1]
	  o XPathNamespace (*Namespace*) [0..*]

Faults	• InvalidChannelFault

5.3 ISBM Notification Services

5.3.1 Notify Listener

Name	Notify Listener
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..*]
	  o Zero only for consumer read response sessions
	• RequestMessageID (*xs:string*) [0..1]
	  o Allows correlation with original request
	  o Only used for consumer read response sessions

Returns	• 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	• ChannelID (*xs:string*) [1]

Behavior	• The ISBM creates a session and returns the SessionID that uniquely identifies the application and channel pair. • If a session already exists (i.e. has not been closed) for the application and channel pair, the existing SessionID is returned. • If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the ChannelType is not a Publication type, then an InvalidChannelTypeFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• InvalidChannelFault • InvalidChannelTypeFault

5.4.2 Post Publication

Name	PostPublication
Description	Posts a publication message on a channel.
Input Parameters	• SessionID (*xs:string*) [1] • PublicationContent (*xs:any*) [1] • TopicName (*xs:string*) [1..*]
Behavior	• The ISBM creates a message with the PublicationContent and a MessageID that uniquely identifies message and then routes the message to the appropriate subscription queues 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. • For any associated topic, if an XPath expression evaluation returns non-valid XML, an InvalidContentFault is thrown.
Returns	• MessageID (*xs:string*) [1]
Faults	• InvalidSessionFault • InvalidTopicFault • InvalidContentFault

5.4.3 Close Publication Session

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

5.5 ISBM Consumer Publication Services

5.5.1 Open Subscription Session

Name	OpenSubscriptionSession
Description	Opens a subscription session for a channel.
Input Parameters	• ChannelID (*xs:string*) [1] • TopicName (*xs:string*) [1..*] • ListenerURI (*xs:anyURI*) [0..1]
Behavior	• The ISBM creates a session and returns the SessionID that uniquely identifies the application and channel pair. • If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the ChannelType is not a Publication type, then an InvalidChannelTypeFault 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.
Returns	• SessionID (*xs:string*) [1]
Faults	• InvalidChannelFault • InvalidChannelTypeFault • InvalidTopicFault

5.5.2 Read Publication

Name	ReadPublication
Description	Returns the first publication 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	• Message (*Message*) [0..1], composed of:
	  ○ MessageID (*xs:string*) [1]
	  ○ PublicationContent (*xs:any*) [1]
	  ○ TopicName (*xs:string*) [1..*]

Faults	• InvalidSessionFault

5.5.3 Remove Publication

Name	RemovePublication
Description	Deletes the first publication 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

5.5.4 Close Subscription Session

Name	CloseSubscriptionSession
Description	Closes a subscription 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 ISBM Provider Request Services

5.6.1 Open Read Request Session

Name	OpenReadRequestSession
Description	Opens a read request session for a channel.
Input Parameters	• ChannelID (*xs:string*) [1] • TopicName (*xs:string*) [1..*] • ListenerURI (*xs:anyURI*) [0..1]
Behavior	• The ISBM creates a session and returns the SessionID that uniquely identifies the application and channel pair. • If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the ChannelType is not a Request type, then an InvalidChannelTypeFault 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.
Returns	• SessionID (*xs:string*) [1]
Faults	• InvalidChannelFault • InvalidChannelTypeFault • InvalidTopicFault

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	• Message (*Message*) [0..1], composed of:
	  ○ MessageID (*xs:string*) [1]
	  ○ RequestContent (*xs:any*) [1]
	  ○ 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 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	• ChannelID (*xs:string*) [1]
Behavior	• The ISBM creates a session and returns the SessionID that uniquely identifies the application and channel pair. • If a session already exists (i.e. has not been closed) for the application and channel pair, the existing SessionID is returned. • If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the Channel Type is not a Response type, then an InvalidChannelTypeFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• InvalidChannelFault • InvalidChannelTypeFault

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] • ResponseContent (*xs:any*) [1]
Behavior	• The ISBM creates a message with the ResponseContent and a MessageID that uniquely identifies message and then routes the message to the appropriate read response queues on the channel. • If the SessionID does not exist then an InvalidSessionFault is thrown. • The ISBM 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 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	• ChannelID (*xs:string*) [1]
Behavior	• The ISBM creates a session and returns the SessionID that uniquely identifies the application and channel pair. • If a session already exists (i.e. has not been closed) for the application and channel pair, the existing SessionID is returned. • If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the ChannelType is not a Request type, then an InvalidChannelTypeFault is thrown.
Returns	• SessionID (*xs:string*) [1]

Faults	• InvalidChannelFault • InvalidChannelTypeFault

5.7.2 Post Request

Name	PostRequest
Description	Posts a request message on a channel.
Input Parameters	• SessionID (*xs:string*) [1] • RequestContent (*xs:any*) [1] • TopicName (*xs:string*) [1..*]
Behavior	• The ISBM creates a message with the RequestContent and a MessageID that uniquely identifies message and then routes the message to the appropriate read request queues 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. • For any associated topic, if an XPath expression evaluation returns non-valid XML, an InvalidContentFault is thrown.
Returns	• MessageID (*xs:string*) [1]
Faults	• InvalidSessionFault • InvalidTopicFault • InvalidContentFault

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 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	• ChannelID (*xs:string*) [1] • ListenerURI (*xs:anyURI*) [0..1]
Behavior	• The ISBM creates a session and returns the SessionID that uniquely

	identifies the application and channel pair. • If the ChannelID does not exist, then an InvalidChannelFault is thrown. • If the Channel Type is not a Response type, then an InvalidChannelTypeFault is thrown.
Returns	• SessionID (*xs:string*) [1]
Faults	• InvalidChannelFault • InvalidChannelTypeFault

5.7.5 Read Response

Name	ReadResponse
Description	Returns the first response 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	• Different to other read services, the returned message contains the originating RequestMessageID for correlation purposes. • If the SessionID does not exist then an InvalidSessionFault is thrown.

Returns	• Message (*ResponseMessage*) [0..1], composed of:
	  ○ MessageID (*xs:string*) [1]
	  ○ RequestMessageID (*xs:string*) [1]
	  ○ ResponseContent (*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 then an InvalidSessionFault is thrown.
Returns	• N/A

Faults	• InvalidSessionFault