OpenO&M
Information Service Bus Model (ISBM)
Specification
Version 1.0

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 which 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 develop one 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. 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:

- does not define how the services are implemented
- does not define the architecture of the supporting application or network service
- does not define any specific underlying communication method

Multiple different implementations are envisioned. The network service will have to include some method for storage or caching of exchanged information, and should probably include some form of guaranteed 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.

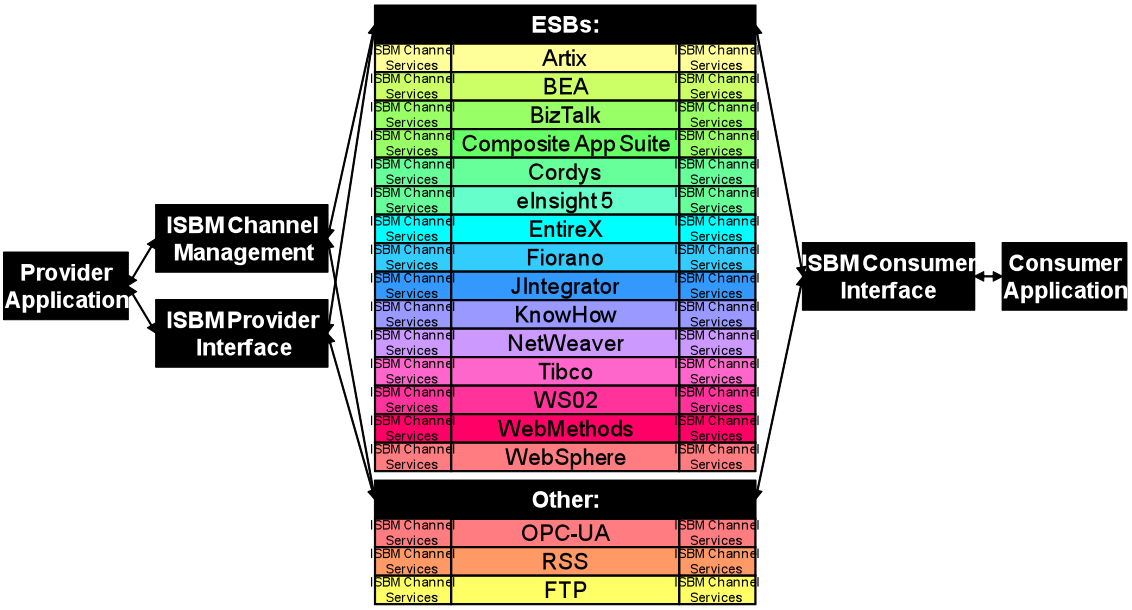

Figure 1 - ISBM Interface to ESB and Other Service Providers

The level of services not defined in this specification, for example type of security, reliability, guaranteed delivery, quality of service, transformation capability, and other features would be provided by the ISBM Service Provider and provide 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

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

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.

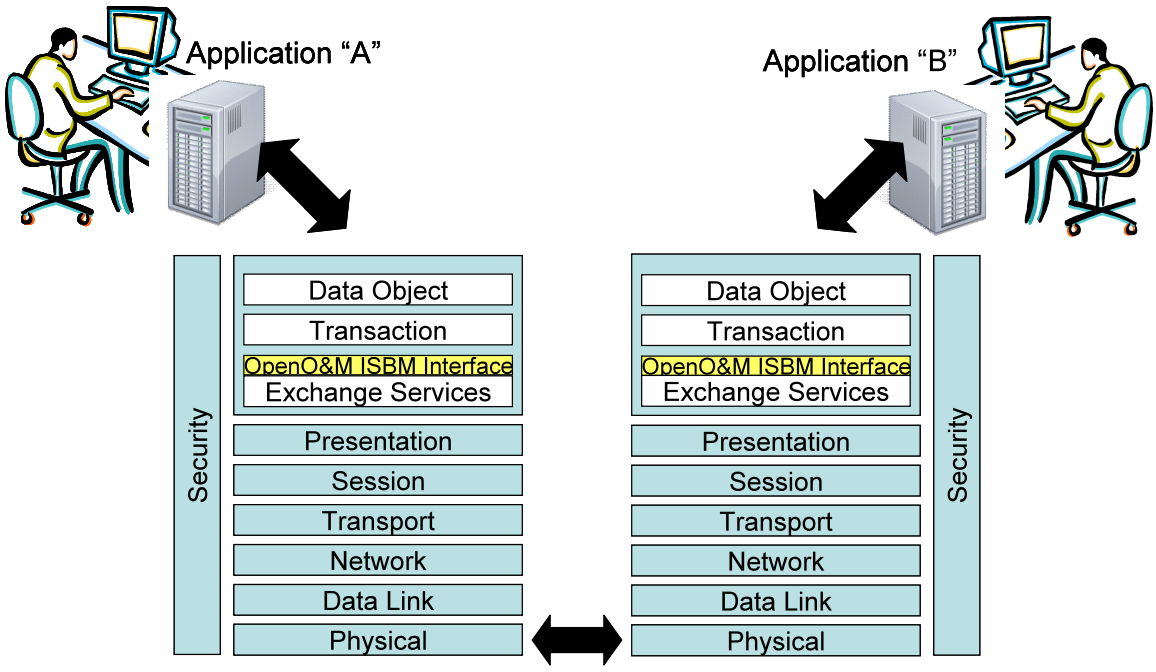

Figure 2 – Loosely Coupled Application Communication Stack

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-ML 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 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.

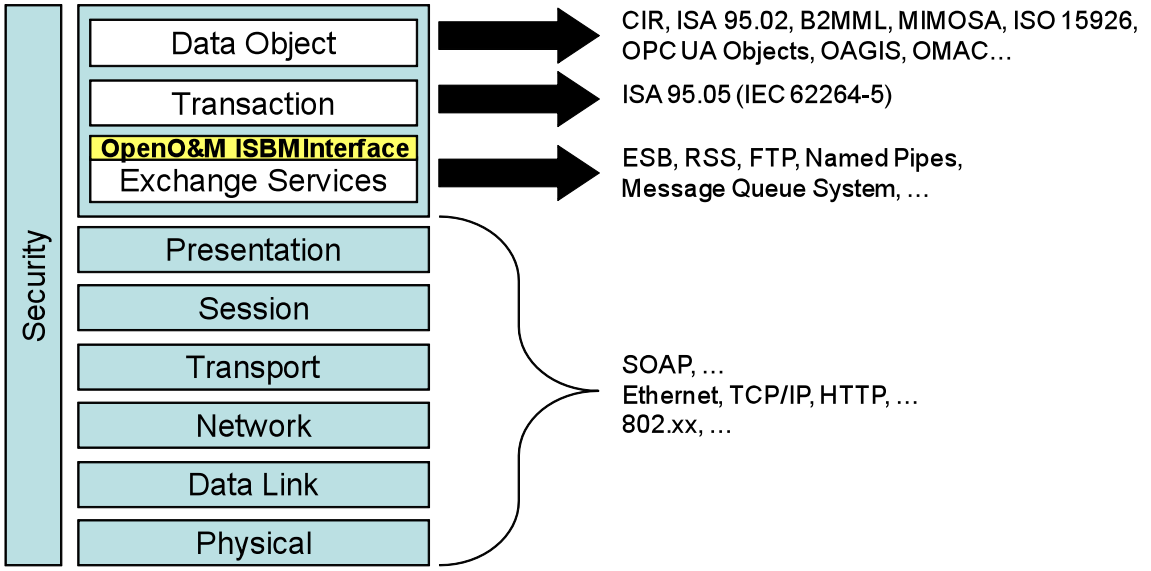

Figure 3 – Defined Standards at Each Application Sublevel

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[2,3].

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)

[2] See the ISA 95 standards for a complete description of the types and format for transactions.

[3] See the WBF B2MML documentation for a complete description of the format and structure for the transactions.

4. Information Sender (to send PROCESS, CHANGE, and CANCEL messages).

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.

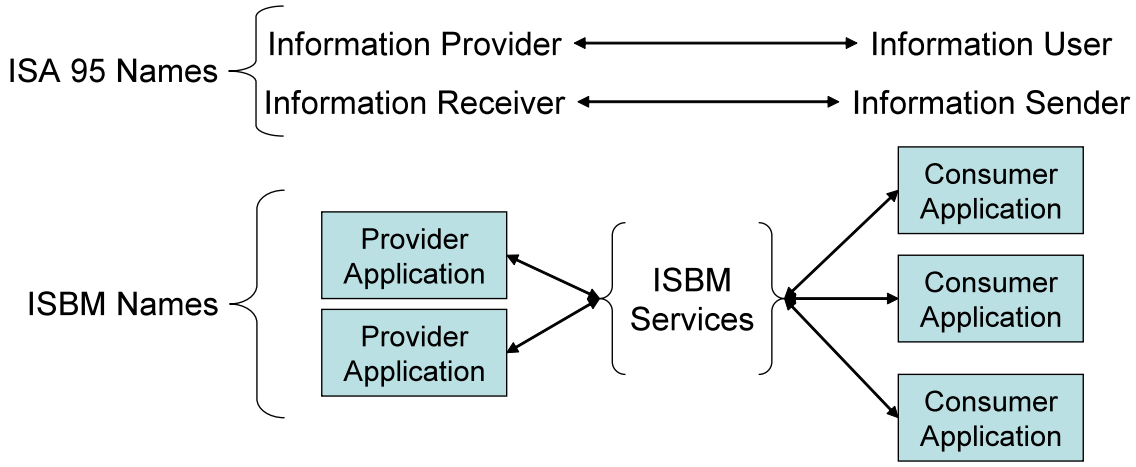

Figure 4 – OpenO&M Information Service Bus Model Names

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 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* is identified using a URI or equivalent identifier. A URI is recommended to allow 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 or request 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 Services

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

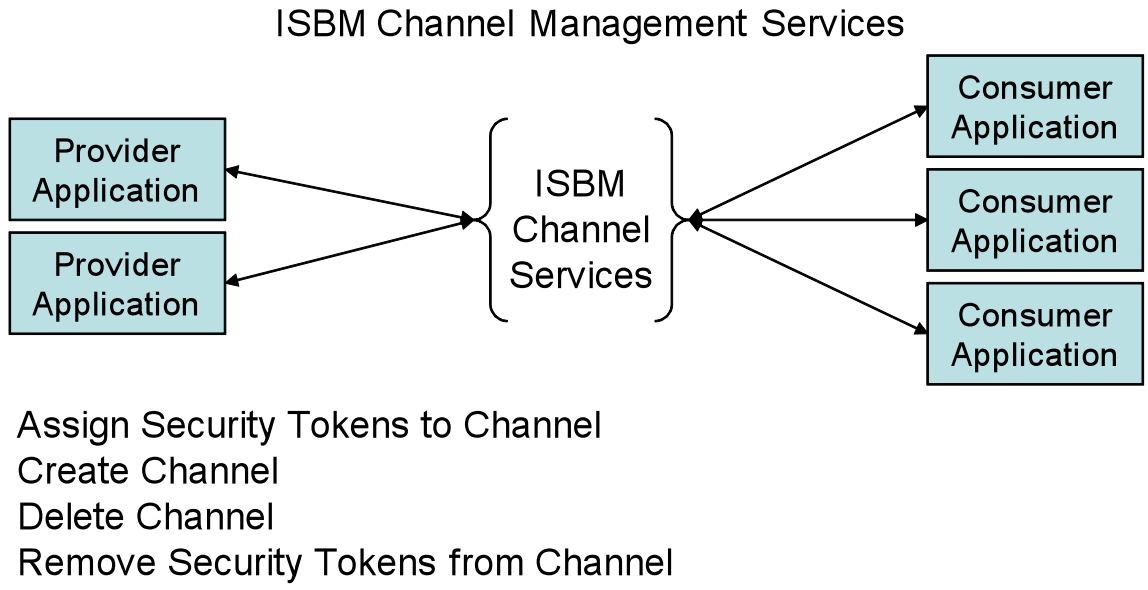

Figure 5 – ISBM Channel Management Services

The ISBM Channel Management Services are used to create and delete channels and to control the security token specification for channels.

2.7 ISBM Publication Channel Services

The ISBM Publication Channel Services are shown in Figure 6. The services allow multiple Provider Applications to post publications to the same channel. Consumer Applications may subscribe to notifications (if supported by the channel) and may read publications.

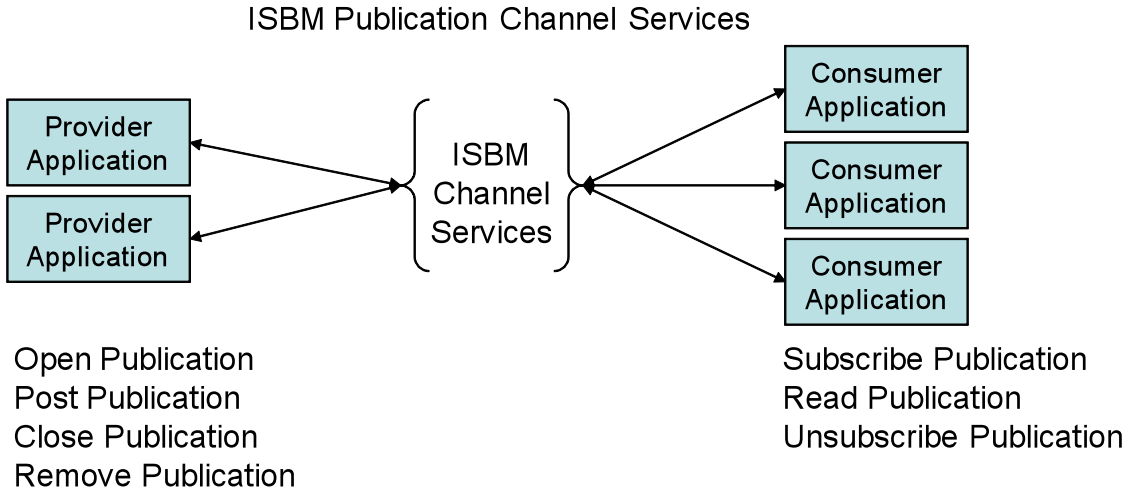

Figure 6 – ISBM Publication Channel Services

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

In this scenario the provider application opens an ISBM publication channel with a channel URI and security token. When the provider application has determined that data should be published it posts publications (using SYNC messages) with a message topic.

A consumer application subscribes to the ISBM publication channel using a channel URI, security token, and 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.

NOTE: See the sections on Methods of Operation for a recommendation of the structure of channels for a more robust actual implementation.

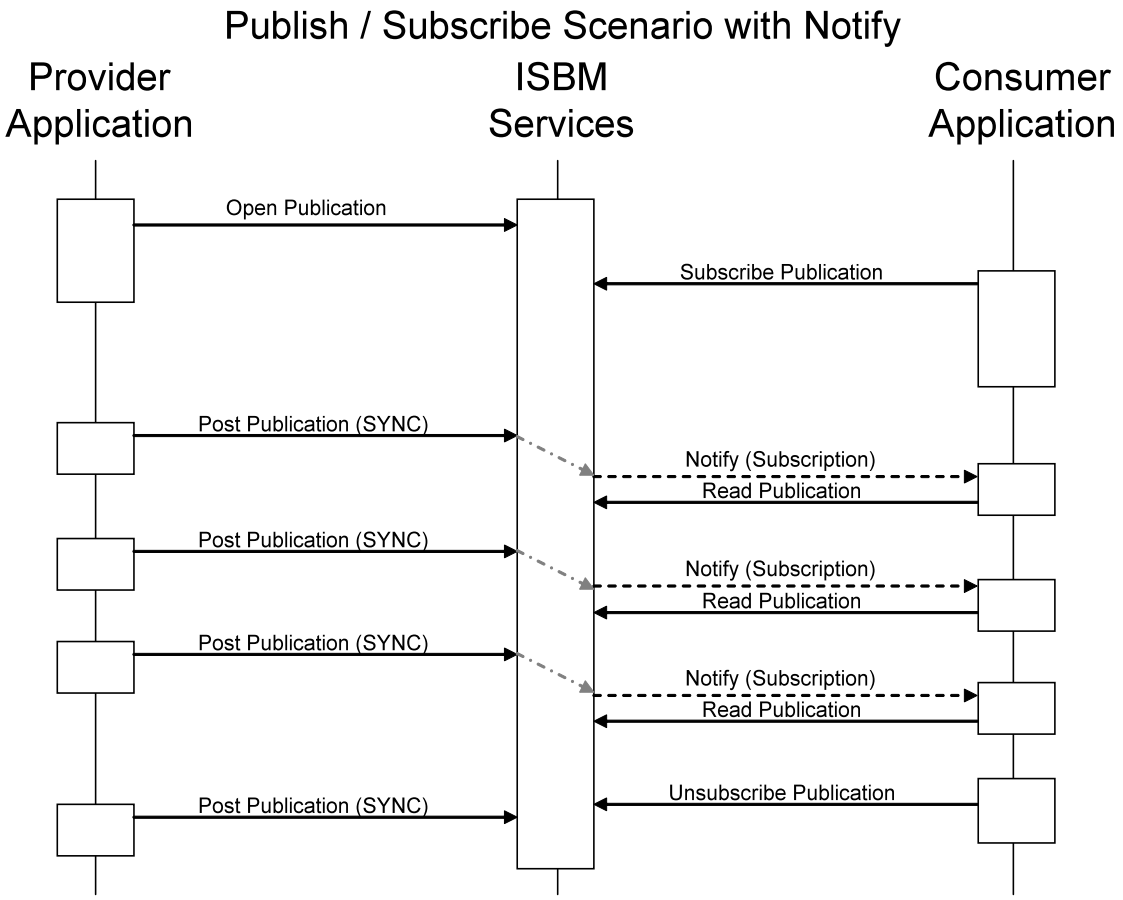

Figure 7 – Publication scenario with notification

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 show 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 indicates if a new publication was returned.

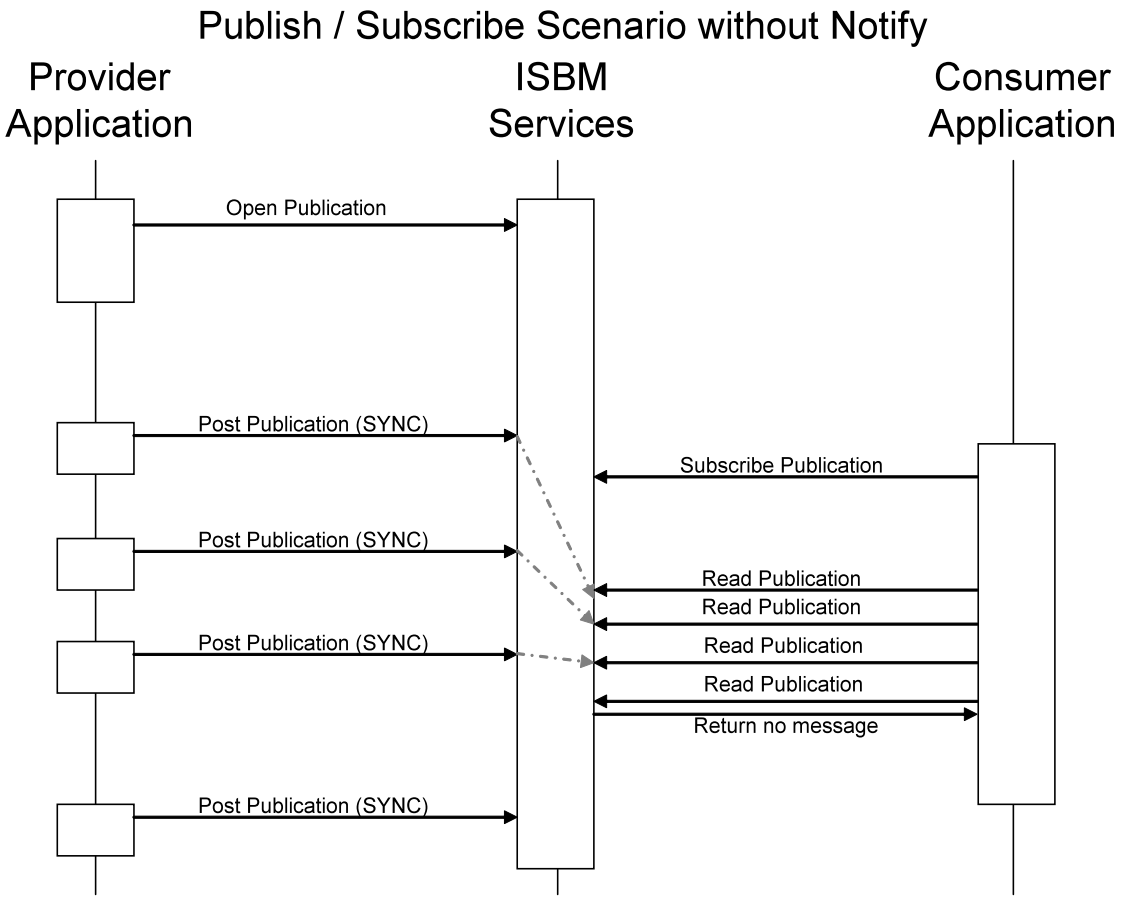

Figure 8 – Publication scenario without notification

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 to Material Definitions while another may publish changes to Material Lots.

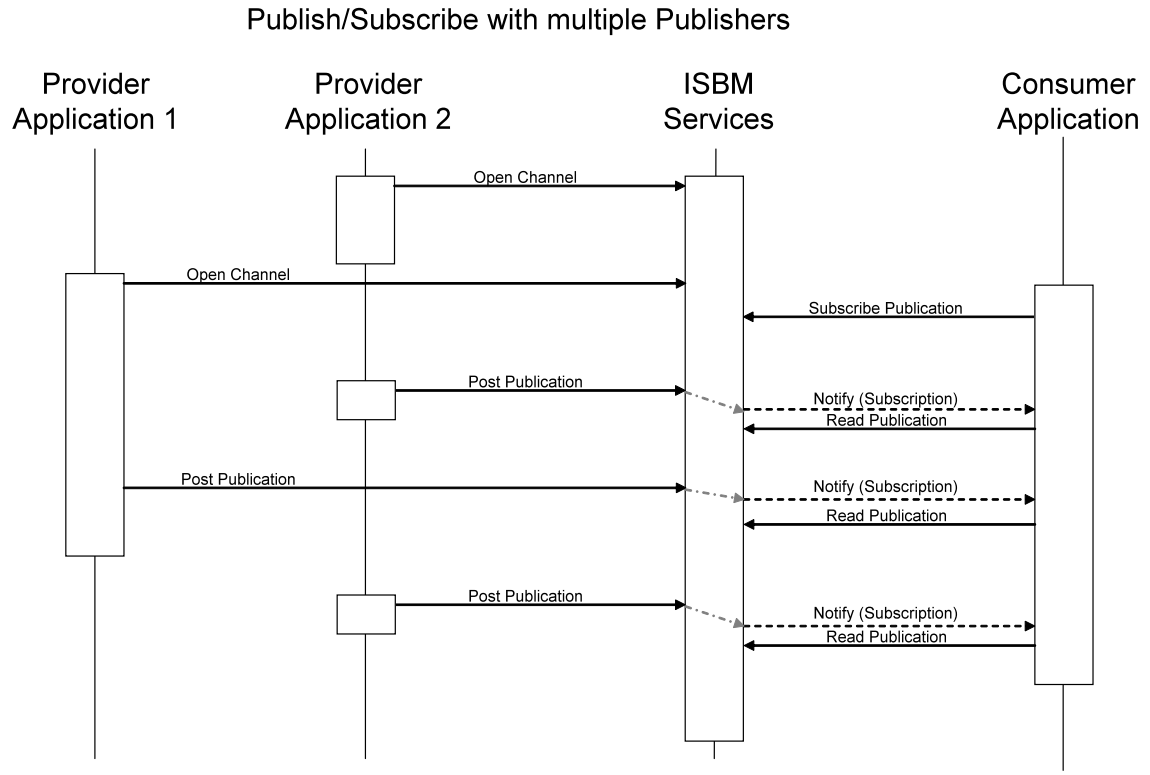

Figure 9 – Publication with multiple provider applications

2.8 ISBM Request Channel Services

The ISBM Channel Services for the ISA 95.05 Push and Pull transactions are shown Figure 10. These are the PROCESS, CHANGE, CANCEL, and GET transactions.

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. Each posted request includes an additional qualifier, called a "Topic", which allows Provider Applications to determine if it should receive the request and post a response to the requestor.

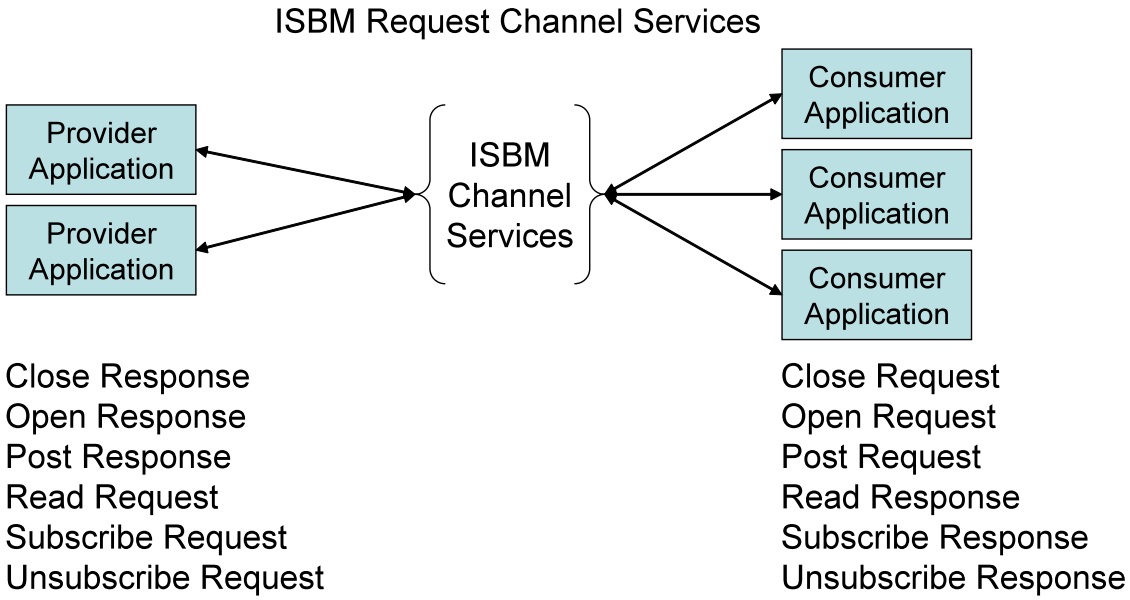

Figure 10 – Services for Request/Response

Figure 11 illustrates a scenario for an ISA 95.05 GET/SHOW transaction with the provider application, consumer application, and ISBM Channel supporting notification.

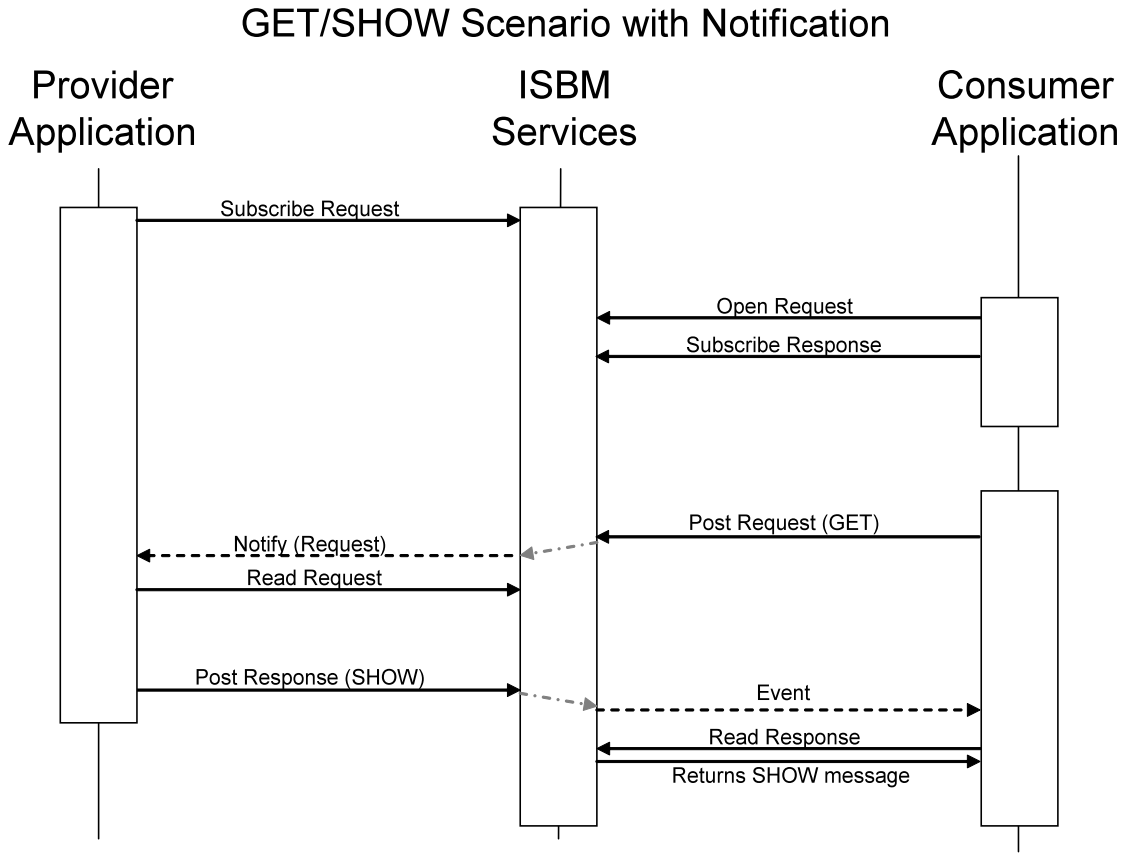

Figure 11 – GET/SHOW Request Service Example

In Figure 11 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 (in this case a SHOW message). The consumer application is notified of the posting and reads the request.

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 scenario using a CHANGE-RESPONSE transaction where the Consumer Application must poll for a response.

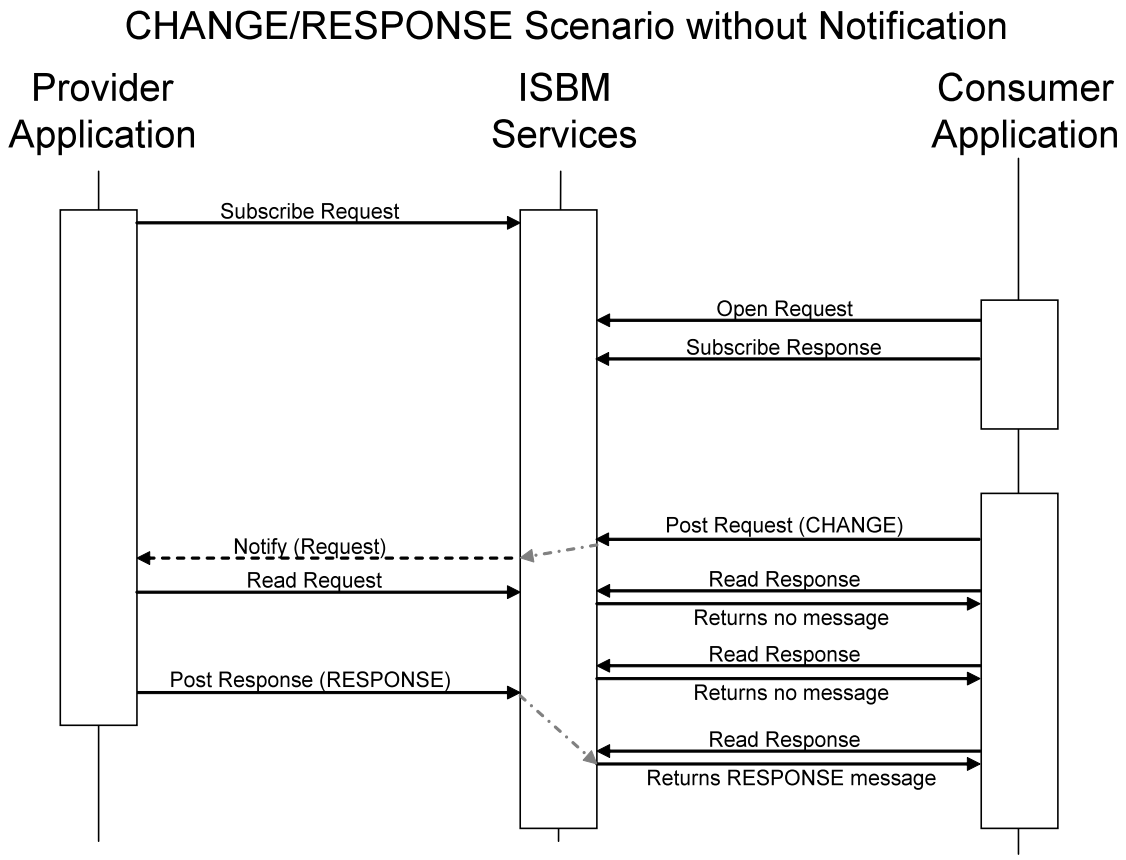

Figure 12 – CHANGE / RESPONSE Request Service Example

The GET/SHOW, PROCESS/ACKNOWLEDGE, CHANGE/RESPONSE, and CANCEL transactions are all handled using request channels.

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 CHANGE request with a specific topic (such as Personnel Information).

Application 1 is notified of a request that matches a topic that it subscribed to. Application 1 gets the CHANGE message and generates the RESPONSE message. 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.

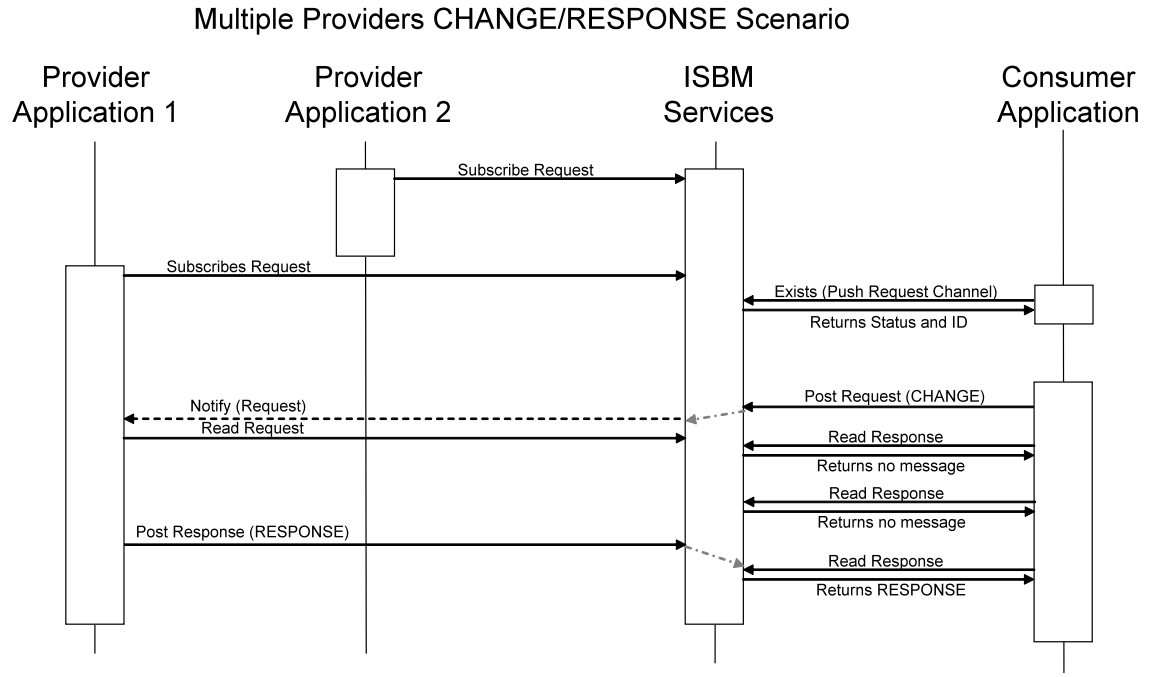

Figure 13 – Multiple Providers CHANGE/RESPONSE Scenario

Note: A full system should not have multiple providers for the same topics on the same request channel. If this occurs then there is a possibility of an indeterminate number of response messages 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 Methods of Operation of ISBM Channels

There is no restriction on the use of Channels and Topics. There are two main elements that should be used for channels and topics, scope of information and type of information.

This section defines a recommended ISBM method for identifying channels IDs and topics that can be used in order to ensure maximum interoperability.

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 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*".

NOTE: The ISBM services do not contain a method to browse the ISBM Roots that are defined. These special services should be provided by an ISBM architecture and may have security restrictions that are outside the scope of the ISBM services.

NOTE: The "<ISBM root>\Request", "<ISBM root>\Checkpoint", and "<ISBM root>\Changes" channels are reserved for browsing applications channels and topics. See Section 5.

3.2 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 company wide 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.

3.3 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 which 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*".

3.4 Channel Use

The following channel use types should be used:

- Channel for requests. The channel use should be "*Request*". This channel would be used for requests from consumer applications to provider applications.
- Channel for publication of changes. The channel use should be "*Publication*". This channel would be used for publication of changes to data from provider applications to consumer applications.

3.5 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.

3.6 Standard Topics

To support interoperability, *topics* will be defined as XPath V1.0 expressions, with a namespace prefix which was previously registered when the topic was defined to the ISBM Provider Application. Standardized namespace prefixes for OpenO&M-supported schemas are:

- B2MML
- CCOM

Examples:

WBF B2MML:

B2MML:PersonnelClass
B2MML:Person
B2MML:QualificationTest
B2MML:EquipmentClass
B2MML:Equipment
B2MML:CapabilityTest
B2MML:MaintenanceRequest
B2MML:MaintenanceWorkOrder
B2MML:MaintenanceResponse
B2MML:MaterialClass
B2MML:MaterialDefinition
B2MML:MaterialLot
B2MML:MaterialSublot
B2MML:QATest
B2MML:ProcessSegment
B2MML:ProductionCapability
B2MML:ProductDefinition
B2MML:ProductionSchedule
B2MML:ProductionPerformance
B2MML:TransactionProfile

MIMOSA CCOM V3:

CCOM:ActualEvent
CCOM:Agent
CCOM:AgentRole
CCOM:AgentRoleWithAgent
CCOM:AgentRoleWithAssessment
CCOM:AgentRoleWithEvent
CCOM:AgentRoleWithMonitoredEntity
CCOM:AgentRoleWithWork
CCOM:Algorithm
CCOM:AlgorithmProcessType
CCOM:AlgorithmTemplate
CCOM:ArrayMeasurement
CCOM:Asset
CCOM:AssetModelHistory
CCOM:AssetOnSegmentHistory
CCOM:AttributeType
CCOM:Audit
CCOM:AverageSynchType
CCOM:AverageType
CCOM:AverageWeightType
CCOM:BinaryObjectMeasurement
CCOM:CalculationType
CCOM:CCOMClass
CCOM:ChangePatternType
CCOM:CompletedWork
CCOM:CPBSetup
CCOM:CRISClass
CCOM:Criticality
CCOM:CriticalityScaleType
CCOM:Cycle
CCOM:DataDictEntry
CCOM:DataQualityType
CCOM:EngineeringStudy
CCOM:EngineeringStudyEntry
CCOM:EngineeringStudyTemplate
CCOM:EngStudyCodeType
CCOM:Enterprise
CCOM:EntityType
CCOM:EntityTypeModelTemplate
CCOM:EnumerationItem
CCOM:EventLink
CCOM:Evidence
CCOM:FFTSetup
CCOM:Function
CCOM:FunctionAffectedByEvent
CCOM:GeoPosition
CCOM:GPSDatumType
CCOM:GPSElevationType
CCOM:GPSLocation
CCOM:GPSPrecisionType
CCOM:GridMeasurement
CCOM:GridTriggeredRegion
CCOM:HealthAssessment
CCOM:HealthAssessmentForEventWithTime
CCOM:HealthLevelType
CCOM:HierarchicalLink
CCOM:HierarchicalNetworkLink
CCOM:HighlightType
CCOM:HypotheticalEvent
CCOM:HypotheticalEventAmbiguitySet
CCOM:HypotheticalEventLogicalConnector
CCOM:InfoSource
CCOM:LogisticResource
CCOM:Manufacturer
CCOM:MaterielItem
CCOM:MaterielMasterItem
CCOM:MeasConnection
CCOM:MeasLocConnection
CCOM:MeasurementLocation
CCOM:MeasurementLocationRegion
CCOM:MeasurementLocationTriggeredRegion
CCOM:MeasurementSource
CCOM:Model
CCOM:Network
CCOM:NetworkConnection
CCOM:OffsetMeasurement
CCOM:OrderedEngineeringStudyItem
CCOM:OrderedList
CCOM:OrderedLogisticResourceItem
CCOM:OrderedMeasLocationWithUnitItem
CCOM:OrderedMeasLocationWithUnitItemSignalProcessSetup
CCOM:OrderedMeasurementLocationItem
CCOM:OSACBMDataType
CCOM:OSACBMFunctionBlockType
CCOM:Port
CCOM:PortIO
CCOM:PossibleEquivalency
CCOM:PostScalingType
CCOM:PriorityLevelType
CCOM:ProposedEvent
CCOM:ProposedEventAmbiguitySet
CCOM:ProposedEventForHypotheticalEvent
CCOM:ProposedEventLogicalConnector
CCOM:PurchaseConditionType
CCOM:ReadinessType
CCOM:Recommendation
CCOM:RecommendationFromEvent
CCOM:RecommendationFromHealthAssessment
CCOM:RecommendationWorkRequest
CCOM:RecommendationWorkStep
CCOM:ReferenceUnitType
CCOM:RegionValidDuringEventType
CCOM:RemainingLifeAssessment
CCOM:Request
CCOM:RequestForEventWithTime
CCOM:RequestForWork
CCOM:Sample
CCOM:ScalarMeasurement
CCOM:Segment
CCOM:SeverityLevelType
CCOM:SignalProcessBlock
CCOM:SignalProcessBlockInStream
CCOM:SignalProcessStream
CCOM:Site
CCOM:SolutionPackage
CCOM:SolutionPackageStep
CCOM:SourceDetectorType
CCOM:SubstantiatedByMeasurement
CCOM:Taxonomy
CCOM:Test
CCOM:TestComponentGroup
CCOM:TestComponentRegion
CCOM:TestComponentTypeRegion
CCOM:TextWithUnitMeasurement
CCOM:TimeWaveformSetup
CCOM:Transducer
CCOM:TransducerAxisDirectionType
CCOM:TriggeredRegionMeasurement
CCOM:TriggeredRegionTestComponent
CCOM:UnitType
CCOM:ValidAttributeTypeForCCOMClass
CCOM:ValidAttributeTypeForEntityType
CCOM:ValidEntityTypeForCCOMClass
CCOM:ValidEntityTypeForEntityType
CCOM:ValidMaterielItemOnSegment
CCOM:ValidModelForMaterielMasterItem
CCOM:ValidNetworkForAgentRoleType
CCOM:ValidNetworkForMonitoredEntity
CCOM:ValidOrderedListForEntityType
CCOM:ValidOrderedListForMonitoredEntity
CCOM:ValidReferenceUnitTypeForMeasLocType
CCOM:ValidTaxonomyForCCOMClass
CCOM:VectorMeasurement
CCOM:VectorRegion
CCOM:VectorSetup
CCOM:WindowType
CCOM:WorkAudit
CCOM:WorkItemType
CCOM:WorkManagementType
CCOM:WorkOrder
CCOM:WorkRequest
CCOM:WorkStep
CCOM:WorkTaskType

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.

3.7 Security

Security in the ISBM services is of paramount importance. In the ISBM Service model 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.

3.7.1 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 or subscribing to a channel. If the application provided security token does not match a security token assigned to the channel, then no channel information is returned.

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.

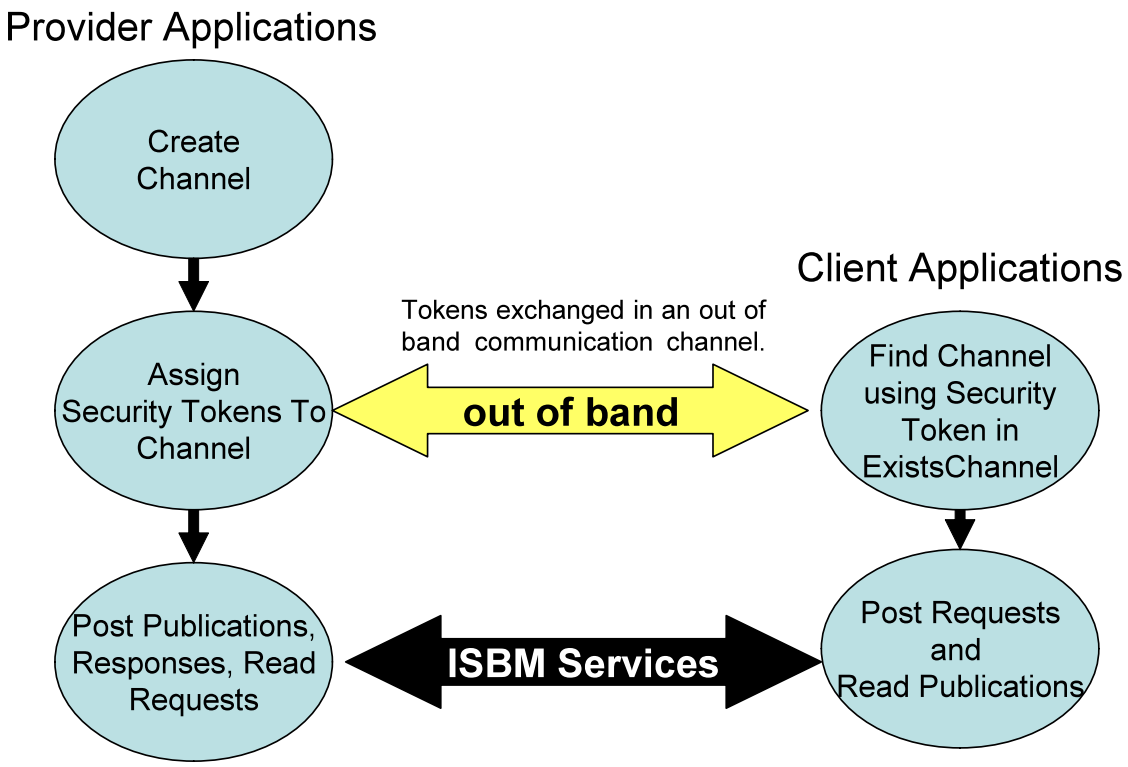

Figure 14 – Security of Channels

3.7.2 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 Provides 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.

The security method is based on the availability of a Security Token Service that can return tokens based upon a request, as shown in Figure 15. The interface to the Security Token Service is not defined as part of this specification.

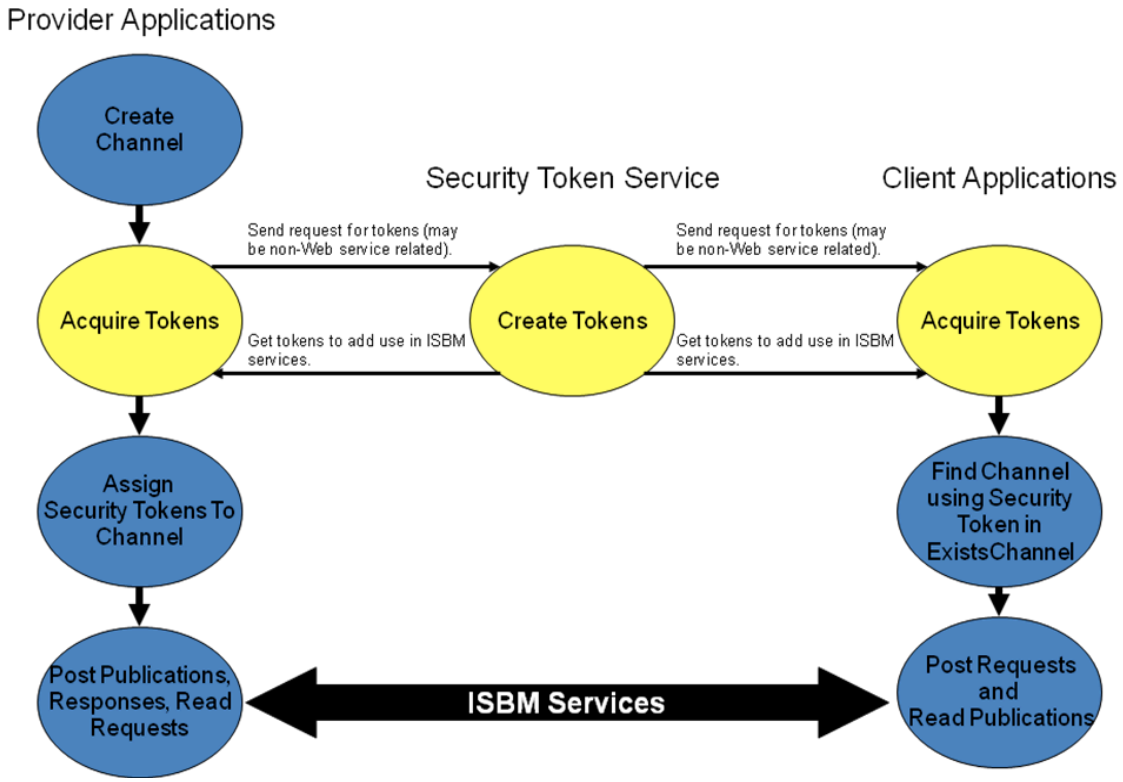

Figure 15 - Security Token Service

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.

3.7.2.1 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.

A password can be passed as plain text or in digest format. An example UsernameToken with a clear text password is:

```
<UsernameToken>
    < Username>Bob Smith</Username>
    < Password Type="PasswordText">hardtobreak</Password>
</UsernameToken>
```

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>
```

3.7.2.2 PKI Through X.509 Certificates

Security tokens may specify an X.509 certificate. 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>
```

3.7.2.3 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>
```

3.7.3 ISBM Service Provider Implementations

1. All *ISBM Service Providers* **must** implement 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 chose to limit the ability to use the ISBM Channel Management services to approved applications, servers, or domains in order to increase security.
4. Provider applications may chose not to apply security tokens to channels. While there is a requirement that the services provide 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.

3.7.4 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, so mechanism should be in setup to tokens on a regular basis.

3. The ISBM services will not validate XML messages. The receiving applications should validate any received messages against the agreed to schema sets, such as B2MML.

3.7.5 ISBM Channel Security Considerations

Some implementations may require additional levels of security than defined in Clause 3.4. For example an implementation may require separate security for GET/SHOW messages than for PROCESS, CHANGE, CANCEL messages. Separate request channels may be setup for the query (GET/SHOW) and process (PROCESS, CHANGE, CANCEL) messages.

3.7.6 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.

3.8 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.

3.8.1 Security Considerations

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

1. The *ISBM Service Provider* may provide an implementation unique method to assign and change security tokens to channels. This removes the need for the provider applications to deal with security issues.
2. 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.
3. 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.
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. Consider having different consumer and provider IDs for messages. If they are the same then a consumer application could remove publications from a publication channel.
6. Consider making Channel IDs very long numbers or strings, non obvious, and not easily guessable in order to prevent access to a channel without going through token security.

3.8.2 Data Format Checking

ISBM Service Providers could provide data format checking 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.

In this situation the ISBM Service Provider could maintain a map between topics and XML schema files. The service provider would use that map to check correctness on posted subscriptions, requests, and responses.

3.8.3 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.

3.8.4 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.

3.8.5 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.

3.8.6 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.

There is no requirement that an *ISBM Service Provider* provide transformation services.

A recommended 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*.

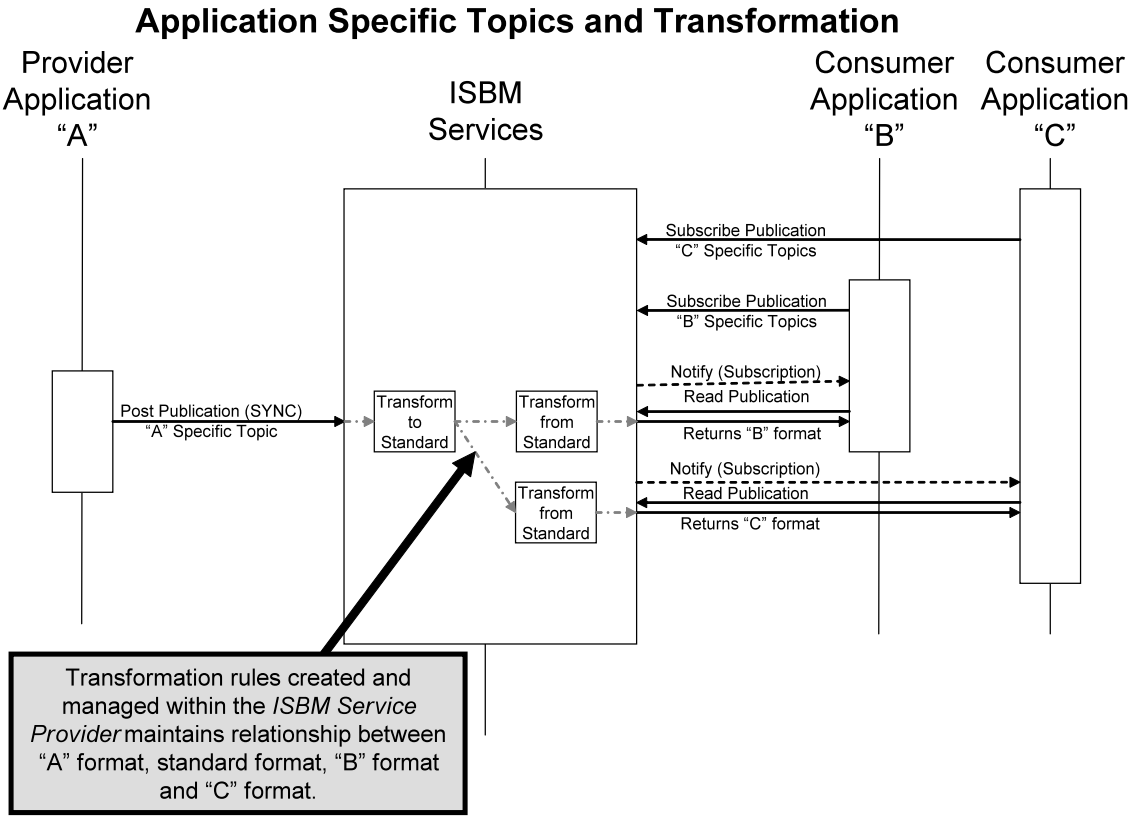

Figure 16 – Transformation Services with the ISBM Service Provider

3.8.7 Cross Company Bridge

ISBM Service Providers could provide cross company communication and authentication services for messages. There is no requirement that an *ISBM Service Provider* provide cross company services.

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 name space to Company B's name space.

The specification of secure or authenticated communication channels is outside the scope of this specification. See WS-STAR specifications for guidance.

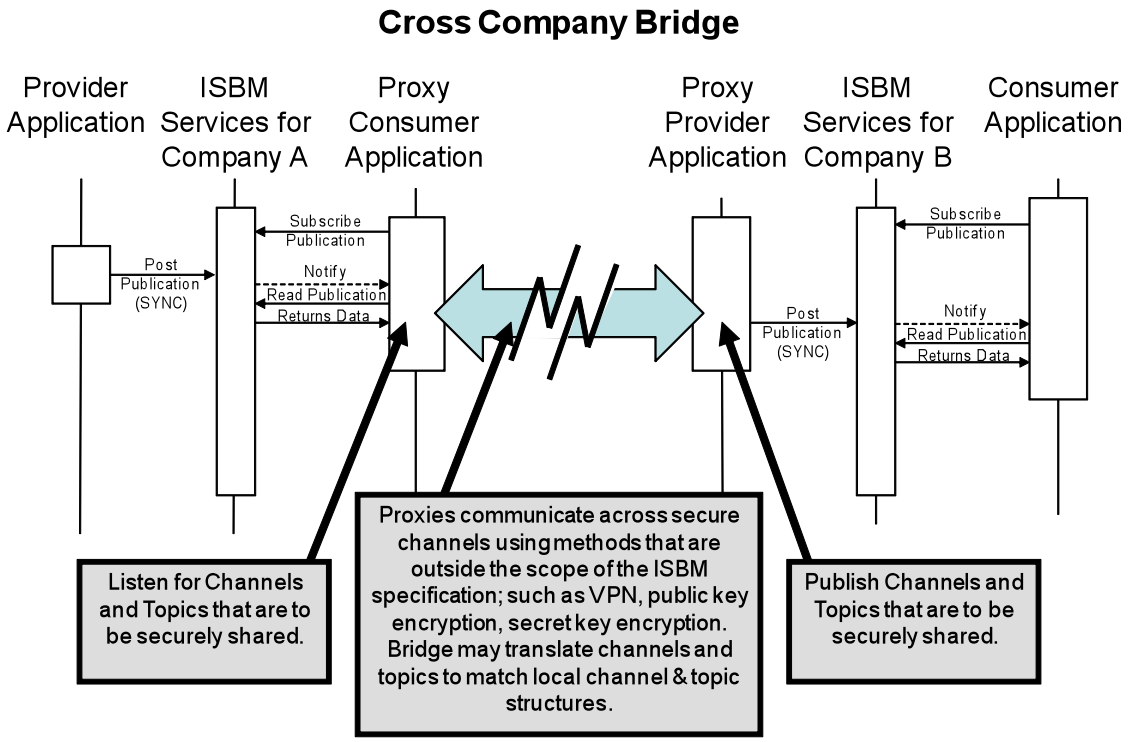

Figure 17 – Cross Company Bridge between ESBs using the ISBM

3.9 W3C Standards

The ISBM services are intended to provide a standard interface, but application infrastructure may already exist to attach to other SOA (Service Oriented Architecture) or web services. In the case where the *ISBM Service Provider* does provide an SOA or web interface, then the ISBM could be exposed using the W3C standards.

Most of the following W3C standards would primarily apply if the ISBM services extended outside the domain of a single organization, or if a single organization used multiple different *ISBM Service Providers*. The specification of communicating *ISBM Service Providers* is beyond the scope of this specification.

3.9.1 WDSL

The Web Services Description Language (WSDL) forms the basis for Web Services.

The *ISBM Service Provider* would describe the ISBM services using WSDL. This definition would be published, probably through directory of services offered by the ESB. The directory should use a Universal Description, Discovery, and Integration (UDDI) registry.

A provider or consumer application would issue one or more queries to the directory to locate the ISBM service and determine how to communicate with that service.

3.9.2 UDDI

A UDDI registry is intended to serve as a means of "discovering" Web Services described using WSDL. Because the ISBM services are standards, the UDDI registry is used to locate the ISBM services.

UDDI is based on a common set of industry standards, including HTTP, XML, XML Schema, and SOAP. The UDDI Business Registry system consists of three directories:

- UDDI white pages: basic information such as a company name, address, and phone numbers, as well as other standard business identifiers like Dun & Bradstreet and tax numbers.
- UDDI yellow pages: detailed business data, organized by relevant business classifications. The UDDI version of the yellow pages classifies businesses according to NAICS (North American Industry Classification System) codes.
- UDDI green pages: information about a company's key business processes, such as operating platform, supported programs, purchasing methods, shipping and billing requirements, and other higher-level business protocols.

UDDI is currently managed by an OASIS technical committee. See http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec for more information.

3.9.3 ebXML Registry

An alternative to UDDI is the ebXML Registry. The ebXML Registry is similar to UDDI in that it allows businesses to find one another, to define trading-partner agreements, and to exchange XML messages in support of business operations. The ebXML goal is to allow all these activities to be performed programmatically without human intervention.

The ebXML initiative is sponsored by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT). See www.ebXML.org for more information.

3.9.4 WS-STAR Standards

WS-STAR (or WS-*) is the name given to the set of services used for SOAP based web services. The services cover how to address services (WS-Addressing), how to achieve reliability (WS-ReliableMessaging), how to describe service capabilities (WS-Policy), and security related standards (WS-Security, WS-SecureConversation, and WS-Trust).

ISBM Service Providers can determine how many of these standards to support and to what extent. Support of the standards would address quality of service issues and integration with other vendor or corporate systems, but would not be visible to ISBM Service users.

4 Service Definitions

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

4.1 Type Definitions

4.1.1 Success and Error Criteria

- 2 = Success But No Message Read

-1 = Success No Notify

0 = Success

1 = Invalid channel ID

2 = Invalid channel URI

3 = Invalid channel type

4.1.2 ISBM Channel Type

1 = Publication Channel Type

2 = Request Channel Type

4.1.3 Channel Name

The Channel name is a string allowing channel names with international character sets.

4.1.4 Channel ID

The channel ID is a string allowing IDs to be encrypted or made non-obvious and not easily guessable.

4.1.5 Channel Session ID

The Channel Session ID is used to identify and manage individual channel consumer/provider sessions and to maintain “last read message” information.

4.1.6 Topic ID

The topic ID is a string, with multiple topics allowed for a channel.

4.2 ISBM Channel Management Services

4.2.1 Assign Security Token

Name	Assign Security Token
Description	Assigns a security token to a channel. Return an error if the channel ID does not match an existing channel.
Input Parameters	▪ Channel ID ▪ Security token
Returns	▪ Success or error criteria ▪ Error message

4.2.2 Create Channel

Name	Create Channel
Description	Creates a new ISBM channel. If the channel does not exist, then the channel is created. If the channel name already exists, then an error is returned. If the provided security tokens do not match an existing token, then an error is returned, otherwise the new security tokens are added to the channel.
Input Parameters	▪ Channel Name ▪ Channel Type (Publication or Request) ▪ List of security tokens
Returns	▪ Channel ID ▪ Success or error criteria ▪ Error message

4.2.3 Create Topic

Name	Create a topic on a channel
Description	Creates a new topic on a channel If the topic does not exist on the channel, then the topic is created. If the topic already exists on the channel, then an error is returned.
Input Parameters	▪ Channel ID ▪ Topic ID ▪ Topic Description ▪ Topic Xpath Definition (Xpath V1.0)
Returns	▪ Success or error criteria ▪ Error message

4.2.4 Delete Channel

Name	Delete Channel
Description	Deletes a channel Returns an error if the channel URI does not match an existing ISBM channel. Any cached or stored messages are no longer available.
Input Parameters	▪ Channel ID
Returns	▪ Success or error criteria ▪ Error message

4.2.5 Delete Topic

Name	Delete Topic
Description	Deletes a topic on a channel Returns an error if the channel ID does not match an existing ISBM channel. Any cached or stored messages are no longer available.
Input Parameters	▪ Channel ID ▪ Topic ID
Returns	▪ Success or error criteria ▪ Error message

4.2.6 Get All Channels

Name	Get All Channels
Description	Returns a list of all accessible channels based on a security token from an ISBM system.
Input Parameters	▪ Security token
Returns	▪ List of Channel ID's ▪ Success or error criteria ▪ Error message

4.2.7 Get All Topics

Name	Get All Topics
Description	Returns a list of all topics on an channel
Input Parameters	▪ Channel ID
Returns	▪ List of Topic ID's ▪ Success or error criteria ▪ Error message

4.2.8 Get Channel Info

Name	Get Channel Information
Description	Returns information about a channel.
Input Parameters	▪ Channel ID
Returns	▪ Channel Name ▪ Channel Type (Publication or Request) ▪ List of Topic ID's ▪ Success or error criteria ▪ Error message

4.2.9 Get Topic Info

Name	Get Topic Information
Description	Returns information about a topic on a channel.
Input Parameters	▪ Channel ID ▪ Topic ID
Returns	▪ Channel ID ▪ Topic ID ▪ Topic Description ▪ Topic Xpath Definition (Xpath V1.0) ▪ Success or error criteria ▪ Error message

4.2.10 Remove Security Token

Name	Remove Security Token
Description	Removes a security token from a channel
Input Parameters	▪ Channel ID ▪ Security token
Returns	▪ Success or error criteria ▪ Error message

4.3 ISBM Notification Channel Services

4.3.1 Notify Listener

Name	Notify Listener
Description	Asynchronous notification of a new message on a channel session based on a previous subscription. This is a callback function invoked by the notifying application based on the Listener URL provided to the notifying application when the application desiring asynchronous notifications subscribed to a channel.
Input Parameters	▪ Channel Session ID ▪ Topic ID ▪ Message ID ▪ Message (Publication, Request, or Response) ▪ Originating Request Message ID (not applicable to Publication Channels)
Returns	▪ Success or error criteria ▪ Error message

4.4 ISBM Consumer Publication Channel Services

4.4.1 Read Publication

Name	Read Publication
Description	For synchronous polling consumers, reads a publication message from a publication channel session message queue based on previous topic subscriptions An application must have previously subscribed to the publication channel and received a channel session ID in order to read publications.
Input Parameters	▪ Channel Session ID ▪ Message ID of last message read
Returns	▪ Topic ID of publication message ▪ Message ID of publication message ▪ Publication message ▪ Success or error criteria ▪ Error message

4.4.2 Subscribe Publication Channel

Name	Subscribe Publication Channel
Description	Subscribes to one or more topics on a publication channel and receives a channel session ID for reading subscriptions synchronously or by providing an asynchronous listener URI
Input Parameters	▪ Channel ID ▪ Security token ▪ URI of Publication Listener ▪ List of Topic ID's
Returns	▪ Channel Session ID ▪ ID of the last read publication message ▪ Success or error criteria ▪ Error message

4.4.3 Unsubscribe Publication Channel

Name	Unsubscribe Publication Channel
Description	Unsubscribes from the publication channel session to stop all messages on all topics and close the session.
Input Parameters	▪ Channel Session ID
Returns	▪ Success or error criteria ▪ Error message.

4.5 ISBM Consumer Request Channel Services

4.5.1 Close Request Channel

Name	Close Request Channel
Description	Closes a request channel from posting of messages.
Input Parameters	▪ Channel Session ID
Returns	▪ Success or error criteria ▪ Error message

4.5.2 Open Request Channel

Name	Open Request Channel
Description	Opens a request channel for posting of request messages. If the channel ID does not exist, or the security token does not match the channel's security token, then no channel session ID is returned.
Input Parameters	▪ Channel ID ▪ Security Token
Returns	▪ Channel Session ID ▪ Message ID of the last posted request message ▪ Success or error criteria ▪ Error message

4.5.3 Post Request

Name	Post Request
Description	Posts a request message to a request channel session
Input Parameters	▪ Channel Session ID ▪ Request message ▪ Topic ID
Returns	▪ Message ID of the request message ▪ Success or error criteria ▪ Error message

4.5.4 Read Response

Name	Read Response
Description	For synchronous polling consumers, reads a response message from a response channel session based on previous response channel subscriptions. An application must have previously subscribed to the response channel and received a channel session ID in order to read responses.
Input Parameters	▪ Channel Session ID ▪ Message ID of last request message read
Returns	▪ Message ID of originating request ▪ Topic ID of response message ▪ Message ID of response message ▪ Response message ▪ Success or error criteria ▪ Error message

4.5.5 Subscribe Response Channel

Name	Subscribe Response Channel
Description	Subscribes to a response channel and receives a channel session ID for reading responses synchronously or by providing an asynchronous listener URI.
Input Parameters	▪ Channel ID ▪ Security token ▪ URI of Response Listener ▪ List of Topics ID's
Returns	▪ Channel Session ID ▪ Message ID of the last read response message ▪ Success or error criteria ▪ Error message

4.5.6 Unsubscribe Response Channel

Name	Unsubscribe Response Channel
Description	Unsubscribes from a response channel session to stop all messages and close the session.
Input Parameters	▪ Channel Session ID
Returns	▪ Success or error criteria ▪ Error message

4.6 ISBM Provider Publication Channel Services

4.6.1 Close Publication Channel

Name	Close Publication Channel
Description	Closes a publication channel session.
Input Parameters	▪ Channel Session ID
Returns	▪ Success or error criteria ▪ Error message

4.6.2 Open Publication Channel

Name	Open Publication Channel
Description	Opens the publication channel for the subsequent posting of publication messages and returns a publication session ID If the channel ID does not exist or the security token does not match the channel's security token, then no channel session ID is returned.
Input Parameters	▪ Channel ID ▪ Security Token
Returns	▪ Channel Session ID ▪ Message ID of the last posted publication message ▪ Success or error criteria ▪ Error message

4.6.3 Post Publication

Name	Post Publication
Description	Posts a publication message to a channel session.
Input Parameters	▪ Channel Session ID ▪ Publication message ▪ Topic ID
Returns	▪ Message ID of posted message ▪ Success or error criteria ▪ Error message

4.6.4 Remove All Publications

Name	Remove All Publications
Description	Removes all published messages from a publication channel session. Returns an error if the channel ID does not match an existing channel.
Input Parameters	▪ Channel Session ID
Returns	▪ Success or error criteria ▪ Error message

4.6.5 Remove Publication

Name	Remove Publication
Description	Removes a specific message from a publication channel session.
Input Parameters	▪ Channel Session ID ▪ Message ID
Returns	▪ Success or error criteria ▪ Error message.

4.7 ISBM Provider Request Channel Services

4.7.1 Close Response Channel

Name	Close Response Channel
Description	Closes a response channel from posting of response messages.
Input Parameters	▪ Channel Session ID
Returns	▪ Success or error criteria ▪ Error message

4.7.2 Open Response Channel

Name	Open Response Channel
Description	Opens a response channel for posting of response messages. If the channel ID does not exist, or the security token does not match the channel's security token, then no channel session ID is returned.
Input Parameters	▪ Channel ID ▪ Security Token
Returns	▪ Channel Session ID ▪ Message ID of the last posted request message ▪ Success or error criteria ▪ Error message

4.7.3 Post Response

Name	Post Response
Description	Posts a response message to a response channel session.
Input Parameters	▪ Channel Session ID ▪ Message ID of originating request ▪ Topic ID of response message ▪ Response message

Returns	▪ Message ID of response message ▪ Success or error criteria ▪ Error message

4.7.4 Read Request

Name	Read Request
Description	For synchronous polling providers, reads a request message from a request channel session based on previous request channel subscriptions. An application must have previously subscribed to the request channel and received a channel session ID in order to read requests.
Input Parameters	▪ Channel Session ID ▪ Message ID of last request message read
Returns	▪ Message ID of request message ▪ Request message ▪ Success or error criteria ▪ Error message

4.7.5 Subscribe Request Channel

Name	Subscribe Request Channel
Description	Subscribes to a request channel and receives a channel session ID for reading requests synchronously or by providing an asynchronous listener URI.
Input Parameters	▪ Channel ID ▪ Security token ▪ URI of Request Listener ▪ List of Topic ID's
Returns	▪ Channel Session ID ▪ ID of the last read request message ▪ Success or error criteria ▪ Error message

4.7.6 Unsubscribe Request Channel

Name	Unsubscribe Request Channel
Description	Unsubscribes from a request channel session to stop all messages and close the session.
Input Parameters	▪ Channel Session ID
Returns	▪ Success or error criteria ▪ Error message

4.8 ISBM WSDL Interface Specifications

The specification includes a Standard WSDL definition for ISBM service providers.

5 Browsing

5.1 Browse Channels, Applications and Topics

The "<ISBM root>\Request", "<ISBM root>\Checkpoint", and "<ISBM root>\Changes" channels are reserved for browsing applications channels and topics.

The reserved "ISBM" channels may be used to access the channel, application, and topic information from the ISBM Services, as if the ISBM Services where a provider application.

AN ISBM *Information Object* is used for the reserved ISBM channels. See 0 for the ISBM Information object XML mapping.

The transaction protocol for ISBM channel information follows the model defined in ISA 95.05 and IEC 62264-5. See these specifications for further information.

The *ISBM Service Provider* should setup security tokens for the reserved channels. .

5.2 ISBM Information Verbs

There are three verbs defined for the ISBM Information objects.

1. GET – Defines a request to the *ISBM Service Provider* to return, in a SHOW message, all sub channels, applications, and topics under the specified channel.
2. SHOW – Defines the returned information from a GET request.
3. SYNC – Defines published data by the *ISBM Service Provider*, for additions, changes, and deletions of channels, applications, or topics.

The GET message may include wildcard specifications for ChannelURIs, ApplicationIDs, and Topics.

On a GET message the ISBM services must provide the list of all channel, application, and topic tuples that match the request.

- Return all Channel URIs that match the passed ChannelURI wildcard specification. To return all Channel URIs use the "*" wildcard in the passed ChannelURI element.
- Return all Provider Application IDs that match the passed Application ID wildcard specification. To return all Application IDs use the "*" wildcard in the passed ApplicationID element.
- Return all Tokens that match the passed Token wildcard specification. To return all token use the "*" wildcard in the passed Token element.

The GET message may contain multiple ServiceElement elements. The GET service shall return tuples matching the above rules for each tuple. Figure 18 illustrates a GET message that can be used to return all channels, provider applications, and topics, for security allowed channels.

```
<?xml version="1.0" encoding="UTF-8"?>
<GetISBMInformation
releaseID=""
xsi:schemaLocation="http://www.openoandm.info/xml/ISBM-V01/ISBMInformation.xsd"
xmlns="http://www.openoandm.info/xml/ISBM-V01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <ApplicationArea>
        <Sender>
            <LogicalID>MyApplication</LogicalID>
        </Sender>
        <CreationDateTime>2001-12-17T09:30:47.0Z</CreationDateTime>
   </ApplicationArea>
   <DataArea>
        <Get/>
        <ISBMInformation>
            <ServiceElement>
                <ChannelURI>"*"</ChannelURI>
                <ApplicationID>"*"</ApplicationID>
                <Topic>"*"</Topic>
            </ServiceElement>
        </ISBMInformation>
   </DataArea>
</GetISBMInformation>
```

Return all channel, attribute, and topic tuples

```
<?xml version="1.0" encoding="UTF-8"?>
<GetISBMInformation
…
   <DataArea>
        <Show/>
        <ISBMInformation>
            <ServiceElement>
                <ChannelURI "ABC" </ChannelURI> <ApplicationID>"DEF"</ApplicationID> <Topic>"GHI"</Topic>
                <ChannelURI "ABC" </ChannelURI> <ApplicationID>"LMN"</ApplicationID> <Topic>"OPQ"</Topic>
                <ChannelURI "RST" </ChannelURI> <ApplicationID>"DEF"</ApplicationID> <Topic>"WXY"</Topic>
            </ServiceElement>
        </ISBMInformation>
   </DataArea>
</GetISBMInformation>
```

Figure 18 – Example of GetISBMInformation and ShowISBMInformation

The following rules should apply to a GET/SHOW transaction:

If only a ChannelURI is specified then
    If Channel Type defined then
        Return only Channel URI records for the specified type
    Else
        Return only Channel URI and channel types records
Else if only ApplicationID is specified then
    If Application Type defined then
        Return only application ID records for the specified type
    Else
        Return only provide application ID records
Else if only Topics is specified then
    Return only Topic records
Else if only ChannelURI and ApplicationID is specified then
    Return only Channel URI and provider application ID records
Else if only ChannelURI and Topic is specified then
    Return only Channel URI and Topic records

Else if only ApplicationID and Topic is specified then

Return only provider application ID and Topic records

Else

Return all ChannelURI, provider application ID, and Topic records

5.3 ISBM Information Schema

The following schema defines the data types to be used for browsing the ISBM Services.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!-- This is a DRAFT    -->

<xsd:schema     xmlns:xsd            = "http://www.w3.org/2001/XMLSchema"
                targetNamespace      = "http://www.openoandm.info/xml/ISBM-V01"
                xmlns                = "http://www.openoandm.info/xml/ISBM-V01"
                elementFormDefault   = "qualified"
                attributeFormDefault = "unqualified">

<!-- Include the Common schema   -->
  <xsd:include schemaLocation = "ISBM-V01-Common.xsd"/>

<xsd:annotation>
   <xsd:documentation>
      This is a draft XML schema used to browse an ISBM service and return
      information that could be used to identify channels and topics using the
      services.
   </xsd:documentation>

   <xsd:documentation>
      Revision History
       Ver     Date            Person          Note
        ---     ----            ------          ----

   </xsd:documentation>
</xsd:annotation>

<!-- Global Elements -->

  <xsd:element name = "ISBMInformation"      type ="ISBMInformationType" />

<!-- Transaction Elements -->
  <xsd:element name = "GetISBMInformation"   type = "GetISBMInformationType"/>
  <xsd:element name = "ShowISBMInformation"  type = "ShowISBMInformationType"/>
  <xsd:element name = "SyncISBMInformation"  type = "SyncISBMInformationType"/>


<!-- Simple & Complex Types  -->

  <xsd:complexType name = "ISBMInformationType">
    <xsd:sequence>
      <xsd:element name = "ServiceElement"
                   type = "ServiceElementType"
                   minOccurs = "0" maxOccurs = "unbounded"/>
      <xsd:element name = "Description"
                   type = "TextType"
                   minOccurs = "0" maxOccurs = "unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
<xsd:complexType name = "ServiceElementType">
  <xsd:sequence>
    <xsd:element name = "ChannelURI"
                 type = "IdentifierType"
                 minOccurs = "0" maxOccurs = "1"/>
    <xsd:element name = "ChannelType"
                 type = "ChannelType"
                 minOccurs = "0" maxOccurs = "1"/>
    <xsd:element name = "ApplicationID"
                 type = "IdentifierType"
                 minOccurs = "0" maxOccurs = "1"/>
    <xsd:element name = "ApplicationType"
                 type = "ApplicationType"
                 minOccurs = "0" maxOccurs = "1"/>
    <xsd:element name = "Topic"
                 type = "IdentifierType"
                 minOccurs = "0" maxOccurs = "1"/>
    <xsd:element name = "LastPostedMessage"
                 type = "TextType"
                 minOccurs = "0" maxOccurs = "1"/>
    <xsd:element name = "LastReadMessage"
                 type = "TextType"
                 minOccurs = "0" maxOccurs = "1"/>
    <xsd:element name = "Description"
                 type = "TextType"
                 minOccurs = "0" maxOccurs = "unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name = "Channel1Type">
  <xsd:simpleContent>
    <xsd:restriction base="CodeType">
      <xsd:enumeration value="Publication"/>
      <xsd:enumeration value="Request"/>
      <xsd:enumeration value="Other"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name = "ChannelType">
  <xsd:simpleContent>
    <xsd:extension base="Channel1Type">
      <xsd:attribute name = "OtherValue" type="xsd:string"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name = "Application1Type">
  <xsd:simpleContent>
    <xsd:restriction base="ApplicationType">
      <xsd:enumeration value="Provider"/>
      <xsd:enumeration value="Consumer"/>
      <xsd:enumeration value="Other"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name = "ApplicationType">
  <xsd:simpleContent>
    <xsd:extension base="Application1Type">
      <xsd:attribute name = "OtherValue" type="xsd:string"/>
    </xsd:extension>
```

```
    </xsd:simpleContent>
  </xsd:complexType>



<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- ISBMInformation Transaction Types    -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

  <xsd:complexType name = "GetISBMInformationType">
    <xsd:sequence>
         <xsd:element name = "ApplicationArea"
                      type = "TransApplicationAreaType"/>
      <xsd:element name = "DataArea">
         <xsd:complexType>
            <xsd:sequence>
              <xsd:element name = "Get"         type = "TransGetType"/>
              <xsd:element name = "ISBMInformation"
                                                type = "ISBMInformationType"
                                                minOccurs = "1"
                                                maxOccurs = "unbounded"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name = "releaseID"
                   type = "xsd:normalizedString"     use="required"/>
    <xsd:attribute name = "versionID"
                   type = "xsd:normalizedString"     use="optional"/>
  </xsd:complexType>


  <xsd:complexType name = "ShowISBMInformationType">
    <xsd:sequence>
      <xsd:element name = "ApplicationArea"
                   type = "TransApplicationAreaType"/>
      <xsd:element name = "DataArea">
         <xsd:complexType>
            <xsd:sequence>
              <xsd:element name = "Show"        type = "TransShowType"/>
              <xsd:element name = "ISBMInformation"
                                                type = "ISBMInformationType"
                                                minOccurs = "1"
                                                maxOccurs = "unbounded"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name = "releaseID"
                   type = "xsd:normalizedString"     use="required"/>
    <xsd:attribute name = "versionID"
                   type = "xsd:normalizedString"     use="optional"/>
  </xsd:complexType>


  <xsd:complexType name = "SyncISBMInformationType">
    <xsd:sequence>
      <xsd:element name = "ApplicationArea"
                   type = "TransApplicationAreaType"/>
      <xsd:element name = "DataArea">
         <xsd:complexType>
            <xsd:sequence>
              <xsd:element name = "Sync"
```

```
                            type = "TransSyncType"/>
                <xsd:element name = "ISBMInformation"
                            type = "ISBMInformationType"
                            minOccurs = "1"
                            maxOccurs = "unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
      </xsd:sequence>
      <xsd:attribute name = "releaseID"
                     type = "xsd:normalizedString"
                     use  = "required"/>
      <xsd:attribute name = "versionID"
                     type = "xsd:normalizedString"
                     use  = "optional"/>
    </xsd:complexType>


</xsd:schema>
```

Annex A: The 7 Layers of the OSI Model

http://www.webopedia.com/quick_ref/OSI_Layers.asp

The OSI, or **O**pen **S**ystem **I**nterconnection, model defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

Application (Layer 7)	This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services. Telnet and FTP are applications that exist entirely in the application level. Tiered application architectures are part of this layer.
Presentation (Layer 6)	This layer provides independence from differences in data representation (e.g., encryption) by translating from application to network format, and vice versa. The presentation layer works to transform data into the form that the application layer can accept. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the syntax layer.
Session (Layer 5)	This layer establishes, manages and terminates connections between applications. The session layer sets up, coordinates, and terminates conversations, exchanges, and dialogues between the applications at each end. It deals with session and connection coordination.
Transport (Layer 4)	This layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.
Network (Layer 3)	This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing.
Data Link (Layer 2)	At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronization. The data link layer is divided into two sub layers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sub layer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronization, flow control and error checking.
Physical (Layer 1)	This layer conveys the bit stream - electrical impulse, light or radio signal -- through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier, including defining cables, cards and physical aspects. Fast Ethernet, RS232, and ATM are protocols with physical layer components.

Annex B: Enterprise Service Buses

The typical IT environment is a federation of systems. The term "federation" in the IT world is applied to collections of applications from multiple vendors that work together to support business processes. A federation may include separate applications for material management, order processing, supply chain management, customer relations, and production scheduling. Even when a company has selected a primary ERP vendor, there is often a federation of legacy systems supporting unique business processes. Federated systems are expensive and integration efforts are often a major portion of IT budgets. An increasingly common method to reduce integration costs is an Enterprise Service Bus (ESB) sometimes called an Enterprise Integration Bus (EIB). These are not electronic busses in the sense of an electrical backplane bus. Instead they are specialized applications that run on redundant servers and act as concentrators and distributors of data. Manufacturing systems that must exchange data with business systems will probably need to connect to the company's ESB.

Enterprise Service Buses are an architectural concept that includes open standards, message based communications, message routing capabilities, and service discovery mechanisms. There is no single definition of an ESB product, but a working rule is that it is a system that provides:

- a single source of shared information,
- a single location for discovering application services, and a
- single destination for using services.

Several vendors are providing ESB, but a few manufacturing companies have also built their own ESB systems based on open standards and focused on their unique integration problems. Once a company has selected an ESB system, then the IT department will attempt to have all applications that exchange data (including manufacturing applications) use the ESB instead of implementing point-to-point connections. Unfortunately, there is little interoperability between different ESB systems, so each application interface must be customized for the chosen ESB.

There are five main elements of ESB's that are important in connecting applications to an ESB;

1. a data transfer element,
2. a service discovery element,
3. a data transform element,
4. a transaction protocol element, and
5. a payload element.

All of the elements are based on XML technologies and newer ESB's are based on web services. The data transfer element handles transporting XML messages from one application to another through the common server. This eliminates point-to-point interfaces and provides a centralized mechanism to manage and view inter-application communication. HTTP messages and JMS (Java Message Services) are common open-source implementations data transfer element layer implementations.

The service discovery element allows applications to discover the services and data provided by the ESB. This is typically handled by UDDI services (www.uddi.org) in the IT environment. In the ISBM, this is the browsing service is provided by the ISBM, but the ISBM services themselves can utilize UDDI.

The data transform element provides methods that convert data from the sender's format to the receiver's format through a set of application specific transform rules. This is often performed

using some form of XML transformation, such as XSLT scripts. This could be handled by the *ISBM Service Provider*.

The transaction protocol element implements the formal definition of allowable message transactions and will utilize the IEC 62264-5 standard.

The payload element defines the data that makes up the body of the message. In the manufacturing area, the standards bodies which compose the OpenO&M Initiative (ISA, WBF, MIMOSA, and OPC) define the XML information payloads.