



IEC 61968-100

Overview
CIM University – UCAIug Summit
Austin, TX

.....

18 November 2011



elster
Vital Connections



Agenda

- Introduction
 - A look at the purpose, scope and key terms and definitions.
- Use Cases and Messaging Patterns
 - What are the key messaging patterns used to exchange data?
- The Message Envelope and Stereotypes
 - How is the data housed for transport and what are the types of messages used?
- Three Flavors of Implementation
 - A look at the JMS, Generic and Strongly-Typed Web Services.



Agenda

- Interoperability Patterns and Use Cases
 - How can the three flavors achieve interoperability in the enterprise?
- Other Considerations
 - Security, WSDL Generation, etc.
- Status
 - Where is this standard now?



Introduction

A look at the purpose, scope and key terms and definitions.



Key Terms and Definitions

- Enterprise Service Bus (ESB)
 - An Enterprise Service Bus (ESB) refers to a software architecture construct that is used as an integration layer. This construct is typically implemented by technologies found in a category of middleware infrastructure products.
 - Typically provides mediation, routing, transformation and logging/auditing logic.
- Java Message System (JMS)
 - A Java Message Oriented Middleware API for sending messages between two or more clients. JMS supports publish/subscribe and point to point messaging patterns



Key Terms and Definitions

- Service Oriented Architecture (SOA)
 - A computer systems architectural style for creating and using business processes packaged as services.
 - SOA separates functions into distinct units (services), which can be distributed over a network and can be combined and reused to create business applications.
- SOAP
 - A protocol specification that contains an envelope for the exchange of structured information via XML.
 - Serves as a foundation layer of the web services protocol stack.



Key Terms and Definitions

- XML Schema (XSD)
 - AN XML document which defines the structure and content of a type of XML document.
 - Can be used to validate XML instance documents.
- Web Services Definition Language (WSDL)
 - An XML document which describes the functionality provided by a web service.
 - Provides a machine-readable description of how the service can be called what parameters it expects and what data structures it returns.



Scope

- IEC 61968
 - Defines architecture (part 1) and interfaces for distribution systems within a utility enterprise.
 - Defines CIM object model and normative XML payloads to exchange CIM derived data. (parts 3 - 9)
- IEC 61968-100
 - Defines profile for application of the other parts of 61968 using common integration technologies, including JMS and web services.
 - Defines normative message envelope and WSDL definitions for interfaces.
 - Provides guidelines and recommendations for usage of Enterprise Service Bus technologies and specific message exchange patterns.
 - Goal: Make IEC 61968 standards more useful in the marketplace.



Purpose

- What is an Implementation Profile?
 - A specification that goes beyond the conceptual level and provides concrete implementation details necessary to produce working software.
- What is the Goal?
 - The other IEC 61968 series of standards provide a conceptual architecture with use cases and normative object payloads. They do not specify how to use specific technologies to produce interoperable software products.
 - The -100 implementation profile is meant to provide concrete message envelopes and interface specifications to enable software to exchange the CIM object payloads using JMS and web services technologies.



Purpose

- Why is this Important?
 - Interoperability.
 - Without these specifications, the distance to integrate applications is much greater.
 - Common message envelope = Less data mapping/transformation requirements.
 - Defined support for message exchange patterns = Less vendor-specific integration code.
 - The results:
 - Lower project costs.
 - Shorter project timelines.
 - Decreased vendor lock-in.

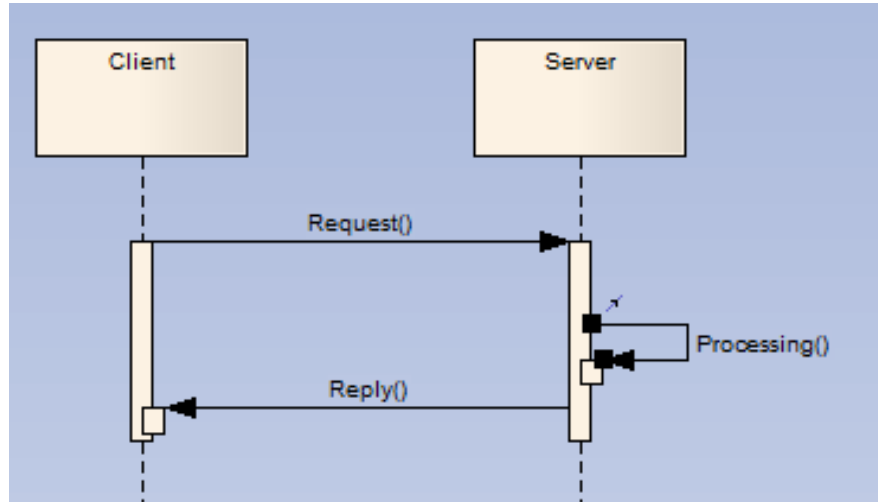


Messaging Patterns

What are the key messaging patterns used to exchange data?

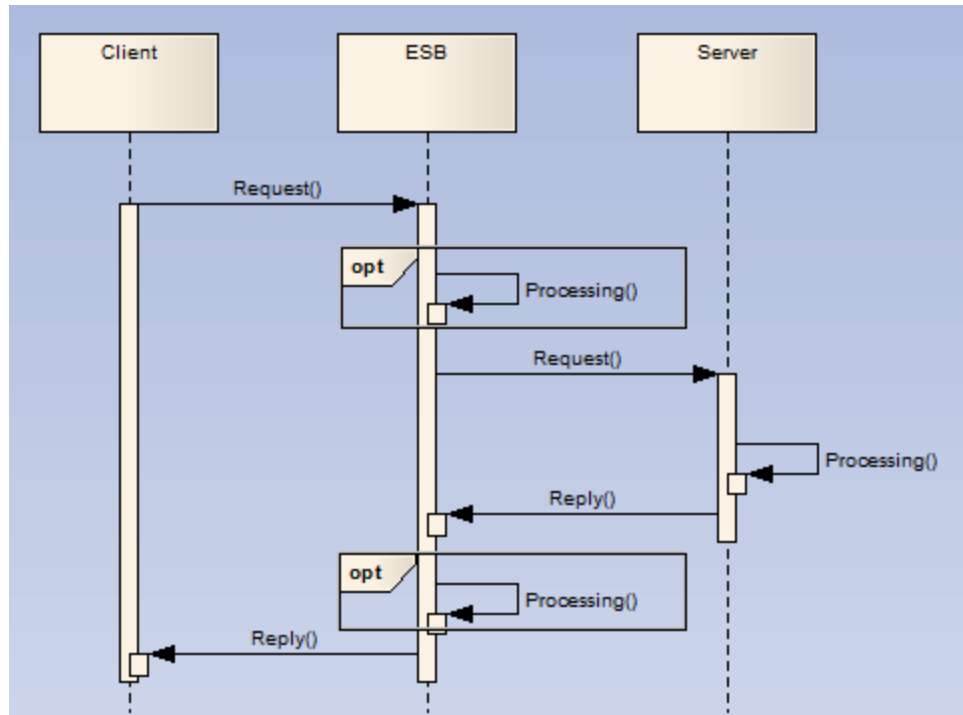


Simple Request/Reply



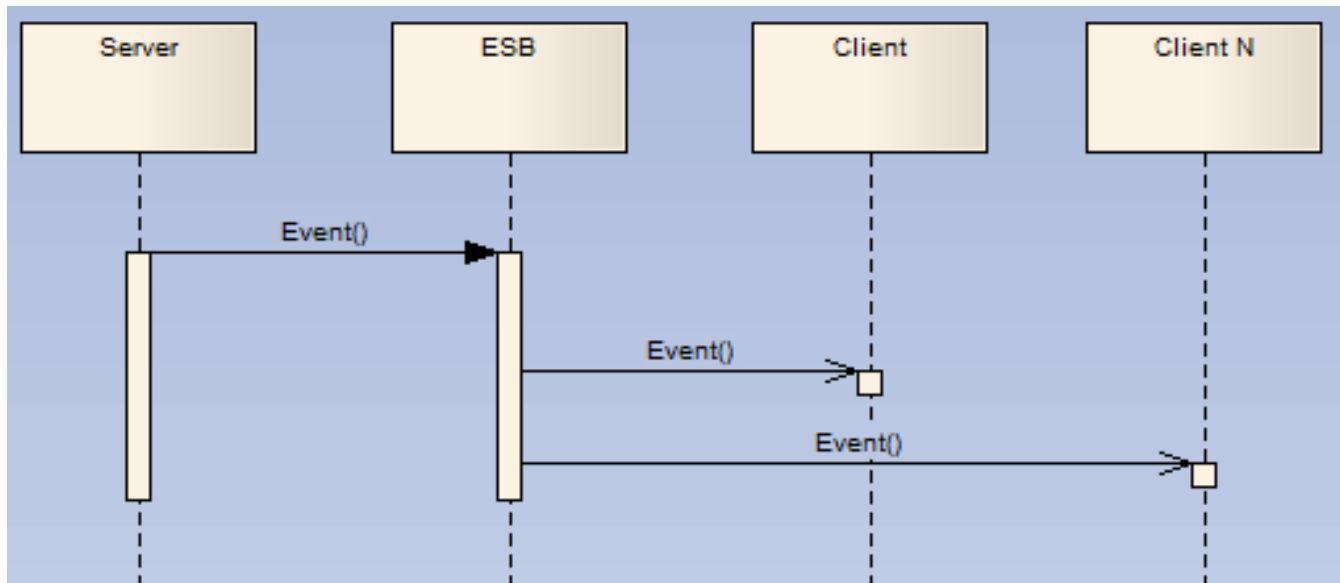
- A client application makes a request to a server.
- The server fulfills the request and reply synchronously with the result.

Request/Reply with an ESB



- A client makes a request to a server via an ESB call.
- The ESB routes the request to the server.
- The server fulfills the request and sends the reply to the ESB.
- The ESB routes the reply back to the client.
- The ESB can provide mediation services in each direction.

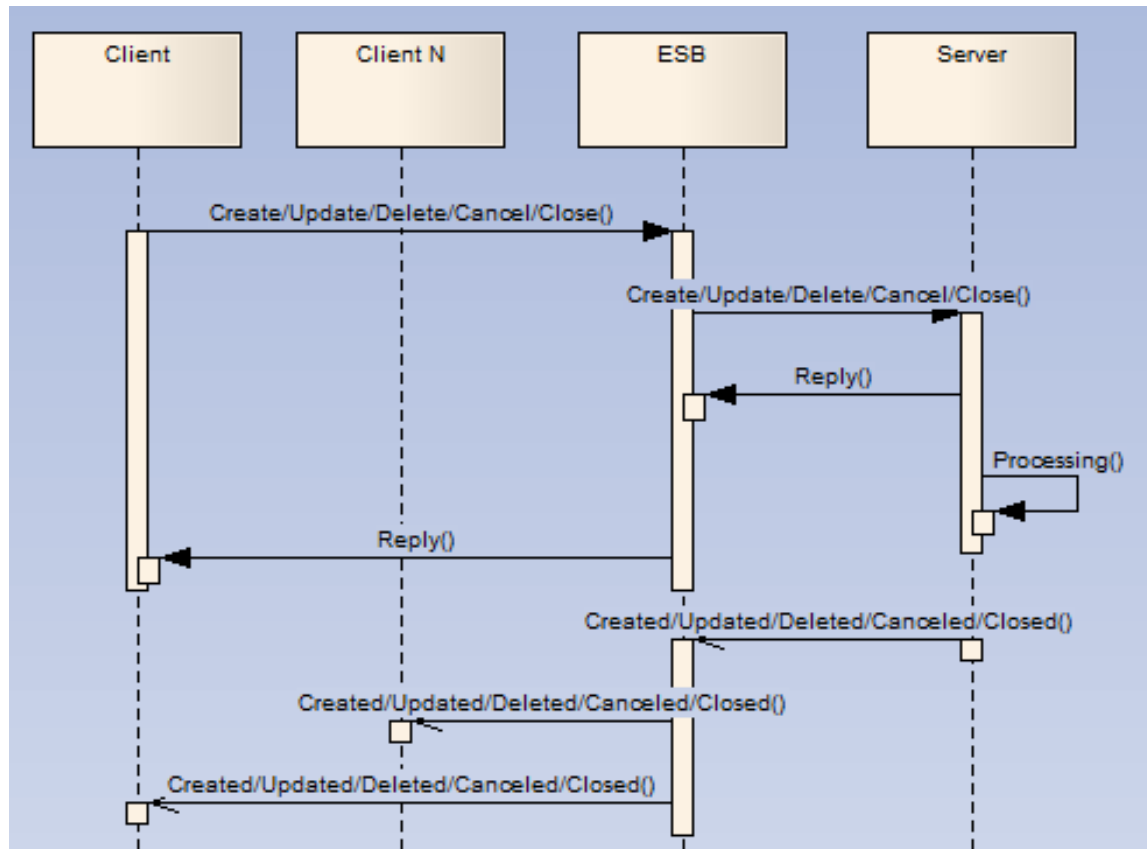
Event Publication



- A server creates and sends an Event to an ESB for publication.
- The ESB routes the event each client application which has an interest in the Event.
- The use of an ESB can provide an enterprise service which contains event propagation knowledge and provides propagation services.
- Isolates individual applications from enterprise business logic.



Complex Messaging



- A combination of request/response and event publication.

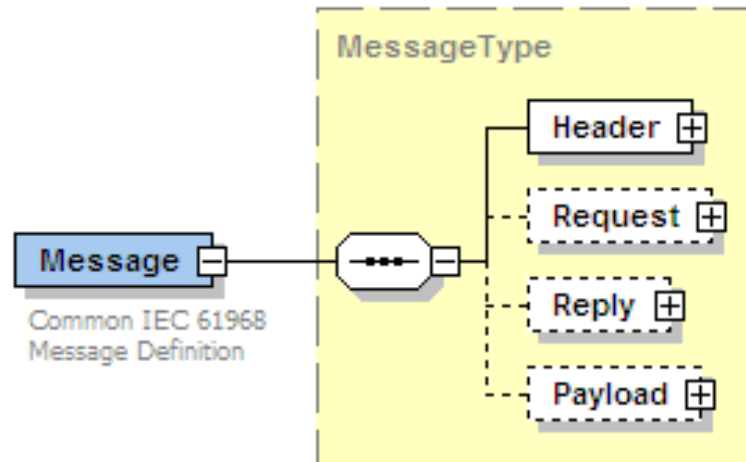


The Message Envelope and Stereotypes

How is the data housed for transport and what are the types of messages used?



Common Message Envelope



- One structure used to convey all 61968 messages.
- Contains the following items:
 - Header: Meta-data about the message. (Verb, Noun, Correlation Id, etc.)
 - Request: Optional request parameters for request messages.
 - Reply: Captures result/error state for response messages.
 - Payload: Core CIM information being exchanged.
- Message stereotypes will have different combinations of these elements.



Header Element

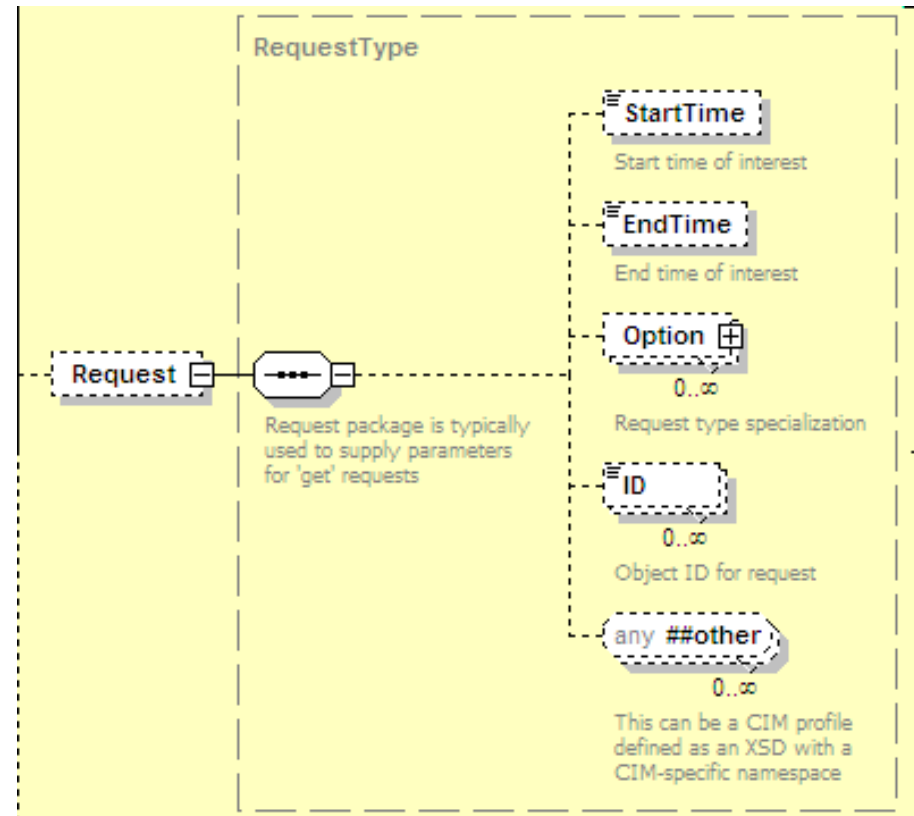


- Required for all messages, except Faults.
- Noun: Identifies the CIM profile being exchanged. Matches Payload.
- Verb: Identifies the action to be taken.
 - create, change, close, cancel, delete + past tense for events (created...)
- Message Id: Unique identifier of the message.
- Correlation Id: Unique identifier used to map related messages.
- Reply Address: Address to specify url for asynchronous replies.
- Revision: Version of standard used. (2.0 for 2nd edition of 61968-9)



Request Element

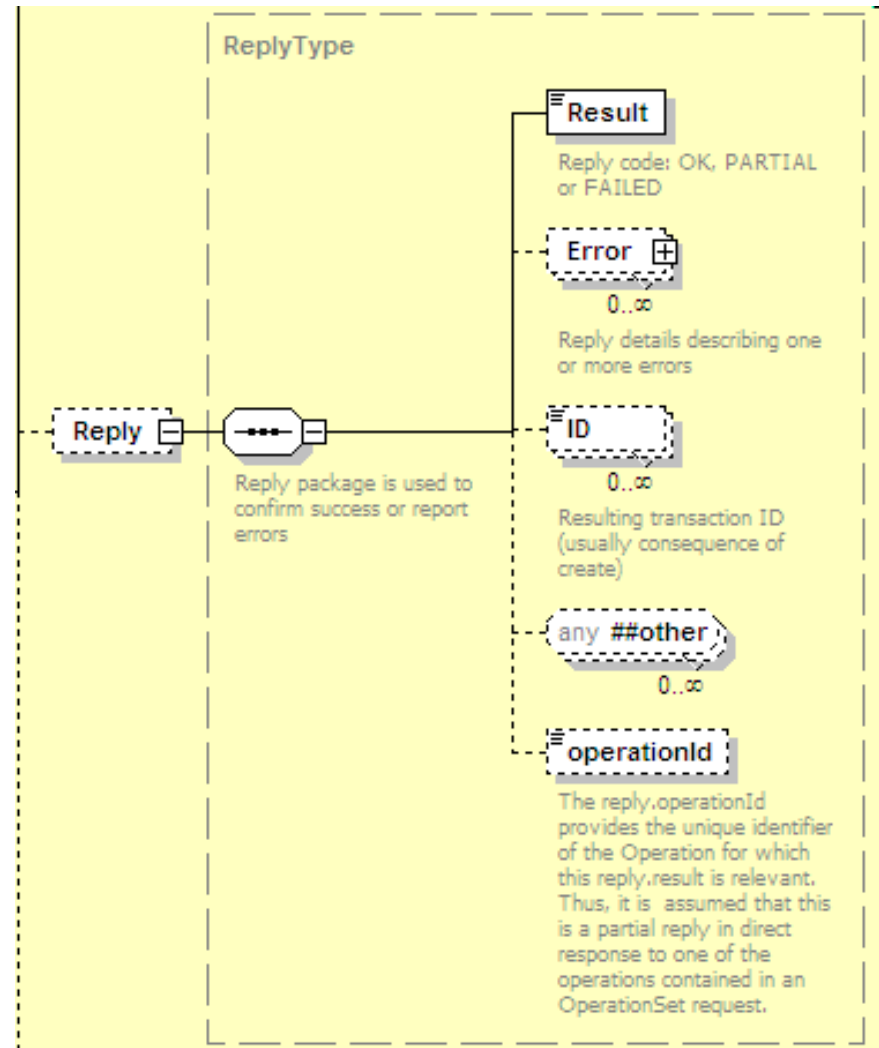
- Used to convey request parameters for request messages.
- Start and End Time: Used for time-based queries.
- Option: Set of name-value pairs.
- xs:any: Location for “Get” profiles.
- Strongly-typed Web Services use specific “Get” profile type here.





Reply Element

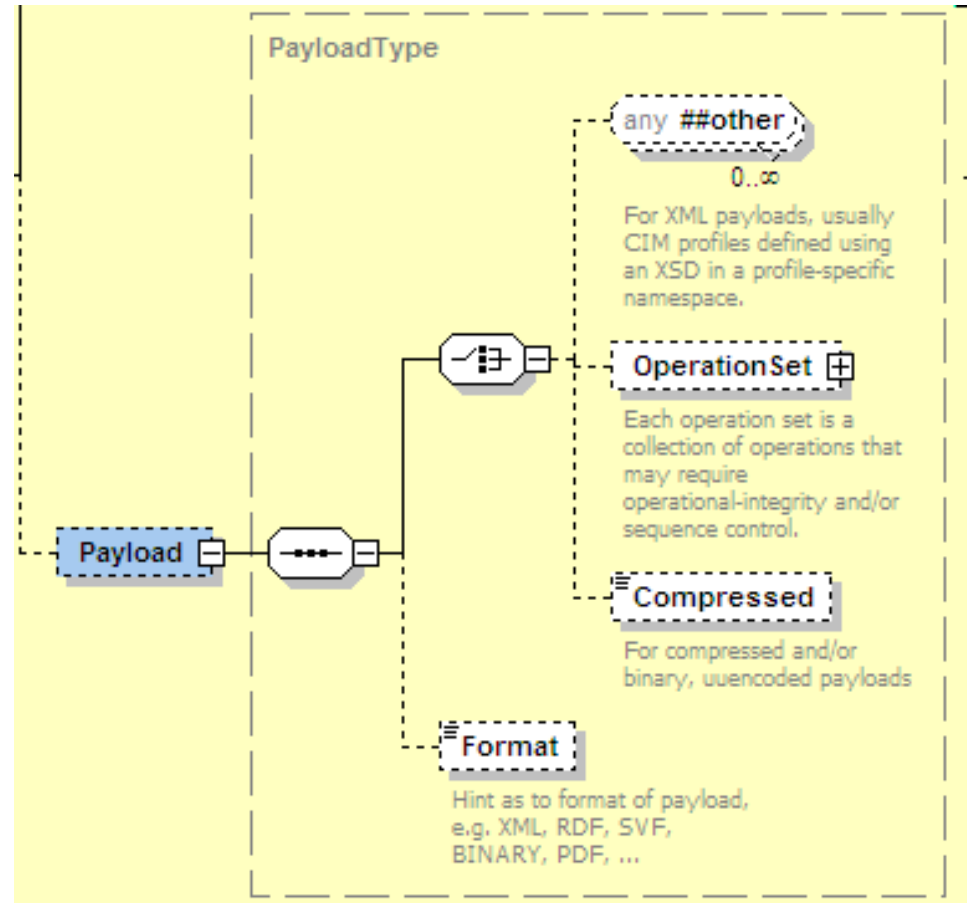
- Captures result/error state for response messages.
- Result: Overall status: OK, PARTIAL, FAILED
- Error: One or more sets of error information.
- operationId: Used to map which operation in an operation set this reply correlated to.





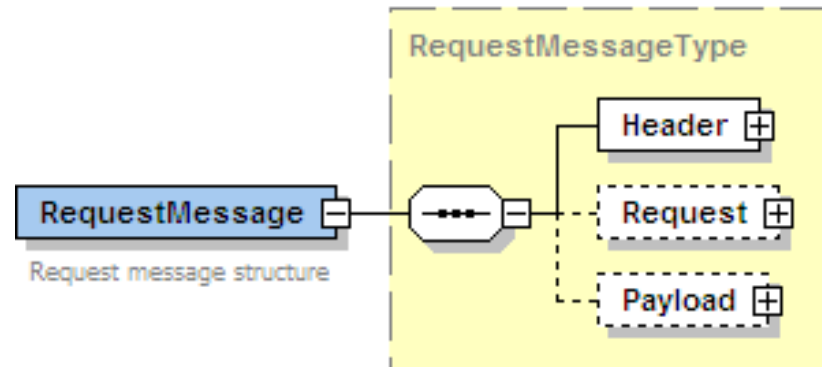
Payload Element

- Holds the CIM profile object being exchanged.
- Matches Noun in Header.
- `xs:any`: XML CIM profile object. Strongly-typed web services use specific payload type here.
- Compressed: Holds compressed payload type for large payloads.
- Format: Format of payload object. (XML, BINARY, RDF, PDF, etc.)



Request Message

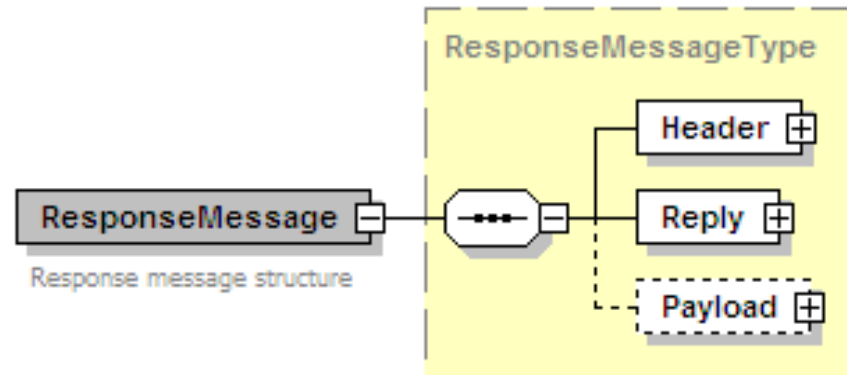
- Used to invoke services to request information or start processing.
- Contains:
 - Header
 - Request (Optional)
 - Payload (Optional)



```
<ns0:RequestMessage xmlns:ns0 = "http://www.iec.ch/TC57/2011/schema/message">
<ns0:Header>
  <ns0:Verb>get</ns0:Verb>
  <ns0:Noun>Switches</ns0:Noun>
  <ns0:Revision>2.0</ns0:Revision>
  <ns0:CorrelationID>1729363b5b7d9c6a0a88d02ae97c64b0</ns0:CorrelationID>
</ns0:Header>
<ns0:Request>
  <ns0:ID>35378383838</ns0:ID>
  <ns0:ID>363482488448</ns0:ID>
  <ns0:ID>894094949444</ns0:ID>
</ns0:Request>
</ns0:RequestMessage>
```

Response Message

- Used to convey the results of a request.
- Contains:
 - Header
 - Reply
 - Payload (Optional)



```
<ns0:ResponseMessage xmlns:ns0 = "http://www.iec.ch/TC57/2011/schema/message">
  <ns0:Header>
    <ns0:Verb>reply</ns0:Verb>
    <ns0:Noun>Switches</ns0:Noun>
    <ns0:Revision>2.0</ns0:Revision>
    <ns0:CorrelationID>1729363b5b7d9c6a0a88d02ae97c64b0</ns0:CorrelationID>
  </ns0:Header>
  <ns0:Reply>
    <ns0:ReplyCode>OK</ns0:ReplyCode>
  </ns0:Reply>
  <ns0:Payload>
    <ns0:Compressed>dghuywqeiwihn353218u23hb2b3b3bhu</ns0:Compressed>
    <ns0:format>XML</ns0:format>
  </ns0:Payload>
</ns0:ResponseMessage>
```

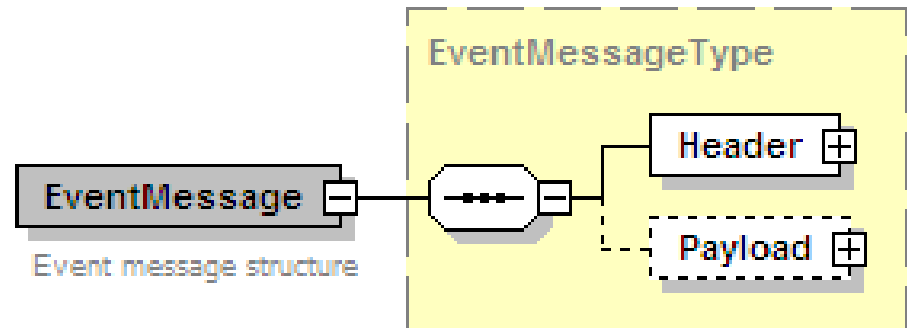


Response Message (Error Example)

```
<ns0:ResponseMessage xmlns:ns0 = "http://www.iec.ch/TC57/2011/schema/message">
  <ns0:Header>
    <ns0:Verb>reply</ns0:Verb>
    <ns0:Noun>Switches</ns0:Noun>
    <ns0:Revision>2.0</ns0:Revision>
    <ns0:CorrelationID>1729363b5b7d9c6a0a88d02ae97c64b0</ns0:CorrelationID>
  </ns0:Header>
  <ns0:Reply>
    <ns0:ReplyCode>ERROR</ns0:ReplyCode>
    <ns0:Error>
      <ns0:level>ERROR</ns0:level>
      <ns0:code>2.8</ns0:code>
      <ns0:details>Unknown object ID: 35378383838</ns0:details>
    </ns0:Error>
  </ns0:Reply>
  <ns0:Payload>
    <m:Switches xsi:schemaLocation="http://www.iec.ch/TC57/2008/CIM-schema-cim12# Switches.xsd" xmlns:m="http://iec.ch/TC57/2007/CIM-schema-cim12#"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <m:Switch>
        <m:mRID>363482488448</m:mRID>
        <m:name>SW2</m:name>
        <m:normalOpen>true</m:normalOpen>
      </m:Switch>
      <m:Switch>
        <m:mRID>894094949444</m:mRID>
        <m:name>SW3</m:name>
        <m:normalOpen>false</m:normalOpen>
      </m:Switch>
    </m:Switches>
  </ns0:Payload>
</ns0:ResponseMessage>
```

Event Message

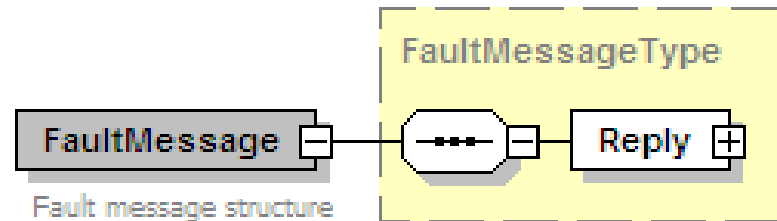
- Published to convey a condition of interest.
- Contains:
 - Header (Past-tense Verb)
 - Payload (Optional)



```
<ns0:EventMessage xmlns:ns0 = "http://www.iec.ch/TC57/2011/schema/message">
  <ns0:Header>
    <ns0:Verb>changed</ns0:Verb>
    <ns0:Noun>Switches</ns0:Noun>
    <ns0:Revision>2.0</ns0:Revision>
  </ns0:Header>
  <ns0:Payload>
    <m:Switches xsi:schemaLocation="http://iec.ch/TC57/2008/CIM-schema-cim12#
Switches.xsd"
      xmlns:m="http://iec.ch/TC57/2007/CIM-schema-cim12#"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <m:Switch>
        <m:mRID>363482488448</m:mRID>
        <m:normalOpen>false</m:normalOpen>
      </m:Switch>
      <m:Switch>
        <m:mRID>894094949444</m:mRID>
        <m:normalOpen>true</m:normalOpen>
      </m:Switch>
    </m:Switches>
  </ns0:Payload>
</ns0:EventMessage>
```

Fault Message

- Reports a failure to process a request message.
 - Malformed or invalid XML.
- Contains:
 - Reply



```
<ns0:FaultMessage xmlns:ns0 = "http://www.iec.ch/TC57/2011/schema/message">
<ns0:Reply>
  <ns0:ReplyCode>ERROR</ns0:ReplyCode>
  <ns0:Error>
    <ns0:level>ERROR</ns0:level>
    <ns0:code>1.8</ns0:code>
    <ns0:details>Malformed XML</ns0:details>
  </ns0:Error>
</ns0:Reply>
</ns0:FaultMessage>
```




Three Flavors of Implementation

A look at the JMS, Generic and Strongly-Typed Web Services.



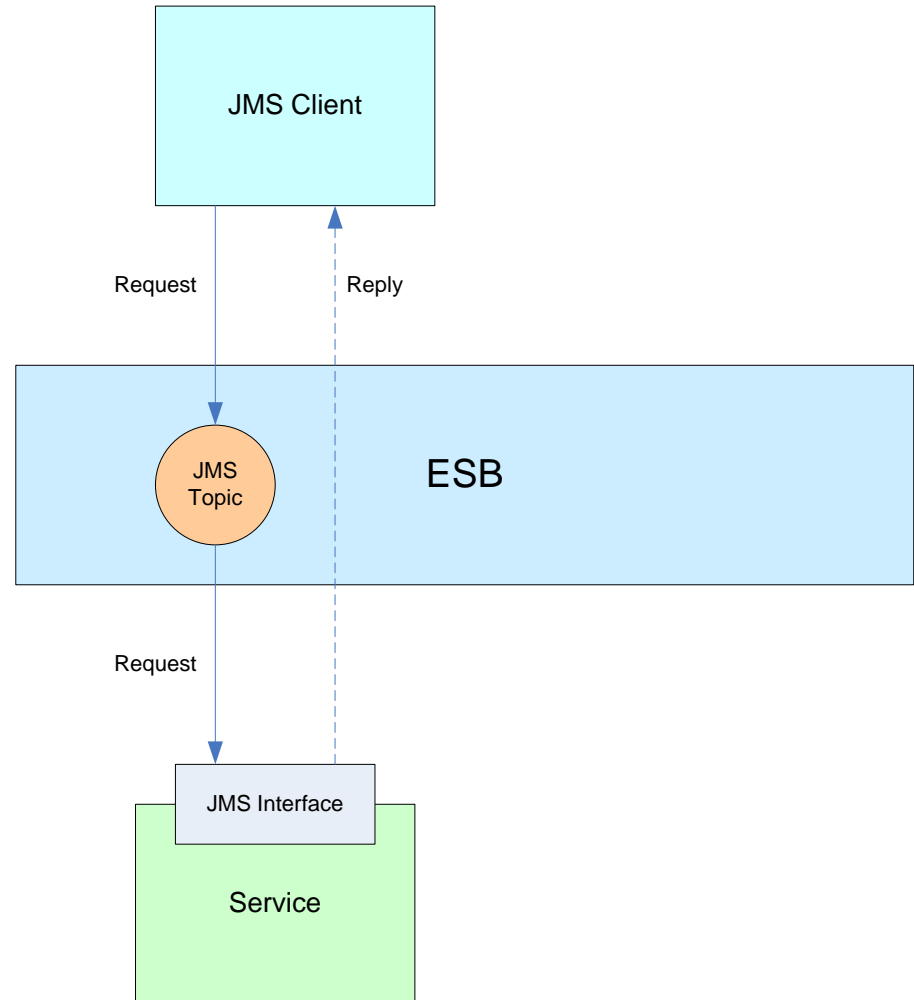
Two Types/Three Favors

- **Web Services (SOAP)**
 - Built on W3C recommended WS-* standards for interface specification and messaging.
 - Interfaces specified via Web Service Definition Language (WSDL).
 - document-literal wrapped
 - WSI Basic Profile 1.1 Compliant
 - Generic-Typed
 - Uses a single WSDL to define all messages.
 - WSDL uses generic payload definition.
 - Strongly-Typed
 - Uses multiple WSDLs to define all messages.
 - SOA approach which defines semantic-based endpoints.
- **Java Message Service (JMS)**
 - Handles exchange of messages using JMS Topics and Queues.
 - Built on JMS apis provided by different messaging middleware vendors.
 - Specifies options around persistent messaging and publish/subscribe mechanisms.



JMS: Request/Reply Pattern

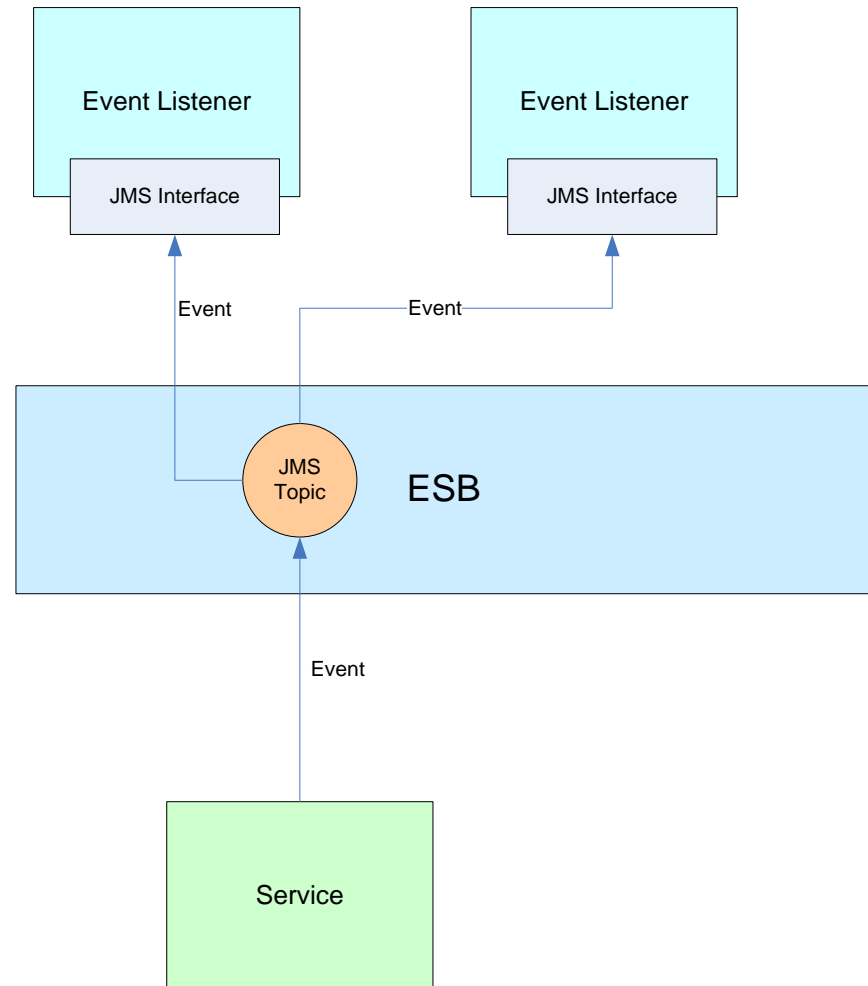
- JMS Queue
 - Destination provided by JMS provider for message buffering. FIFO Queue.
- Request is sent to the appropriate JMS destination.
 - **queue:Request.EndDeviceControls**
- Optional reply is synchronous or asynchronous.





JMS: Event Publishing Pattern

- JMS Topic
 - Destination provided by JMS provider for message publishing. Sent to all subscribers.
- Applications interested in types of events subscribe to a JMS topic.
- Event is sent to the appropriate JMS topic.
 - **topic:Event.EndDeviceControls**
- Topic sends the event message to all subscribers.



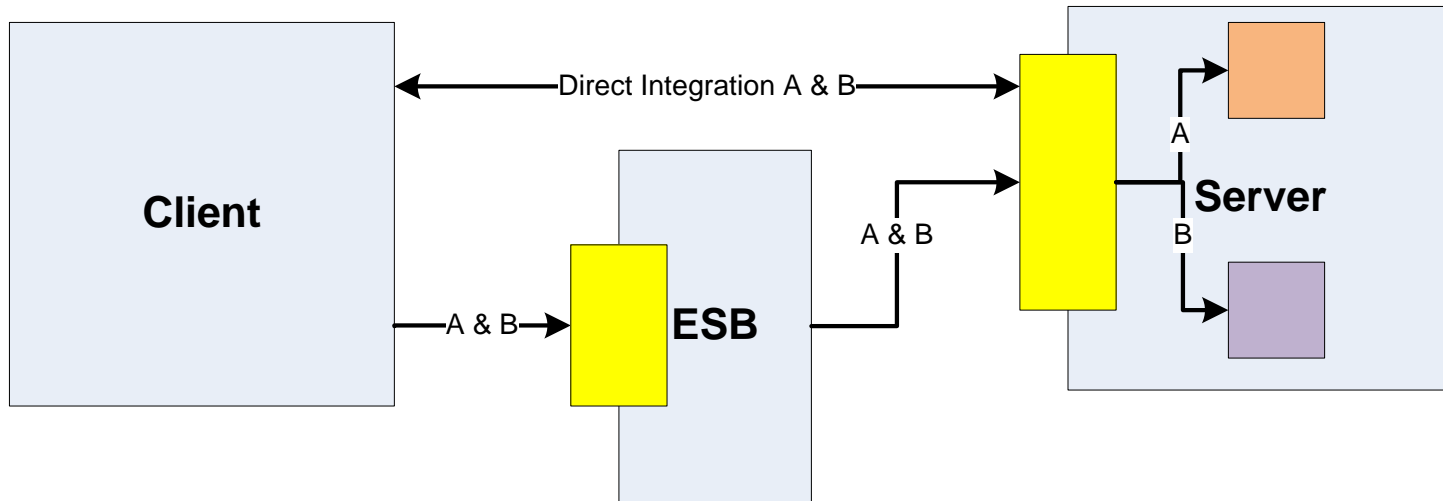


JMS: Message

- Example Request Message:

```
<ns2:RequestMessage xmlns="http://iec.ch/TC57/2011/schema/message"
  xmlns:ns3="http://iec.ch/TC57/2011/EndDeviceControls#">
  <Header>
    <Verb>create</Verb>
    <Noun>EndDeviceControls</Noun>
    <Revision>2.0</Revision>
    <Timestamp>2011-11-10T16:26:54.232Z</Timestamp>
    <Source>EIServer</Source>
    <ReplyAddress>queue:Event.EndDeviceControls</ReplyAddress>
    <MessageID>0d39be6c-ddb0-4056-b9e7-5f15d91658cc</MessageID>
    <CorrelationID>e3f37e15-10e6-43de-8b45-6537022f9065</CorrelationID>
  </Header>
  <Payload>
    <ns3:EndDeviceControls>
      <ns3:EndDeviceControl>
        <ns3:EndDeviceControlType ref="3.31.0.23"/>
        <ns3:EndDevices>
          <ns3:mRID>cf14ac6f-4829-4d32-bbc4-ed90ddfa2760</ns3:mRID>
        </ns3:EndDevices>
      </ns3:EndDeviceControl>
    </ns3:EndDeviceControls>
  </Payload>
</RequestMessage>
```

Generic-Typed Web Services



- Uses a single WSDL to define messages corresponding to the Message Stereotypes.
- Payload is generic (xs:any)
- Typically one endpoint to handle and route requests.
- Introspection of message needed to route appropriately.
- Payload validation is typically application specific.



Generic-Typed: Operations

- Three operations:
 - Request
 - Response
 - PublishEvent
- Type of payload can be inferred from <noun> in header.
- Action to be taken can be inferred from <verb> in header.

```
<wsdl:operation name="PublishEvent">
  <wsdl:input message="ns:EventMessage"/>
  <wsdl:output message="ns:ResponseMessage"/>
</wsdl:operation>
<wsdl:operation name="Request">
  <wsdl:input message="ns:RequestMessage"/>
  <wsdl:output message="ns:ResponseMessage"/>
</wsdl:operation>
<wsdl:operation name="Response">
  <wsdl:input message="ns:ResponseMessage"/>
  <wsdl:output message="ns:ResponseMessage"/>
</wsdl:operation>
```

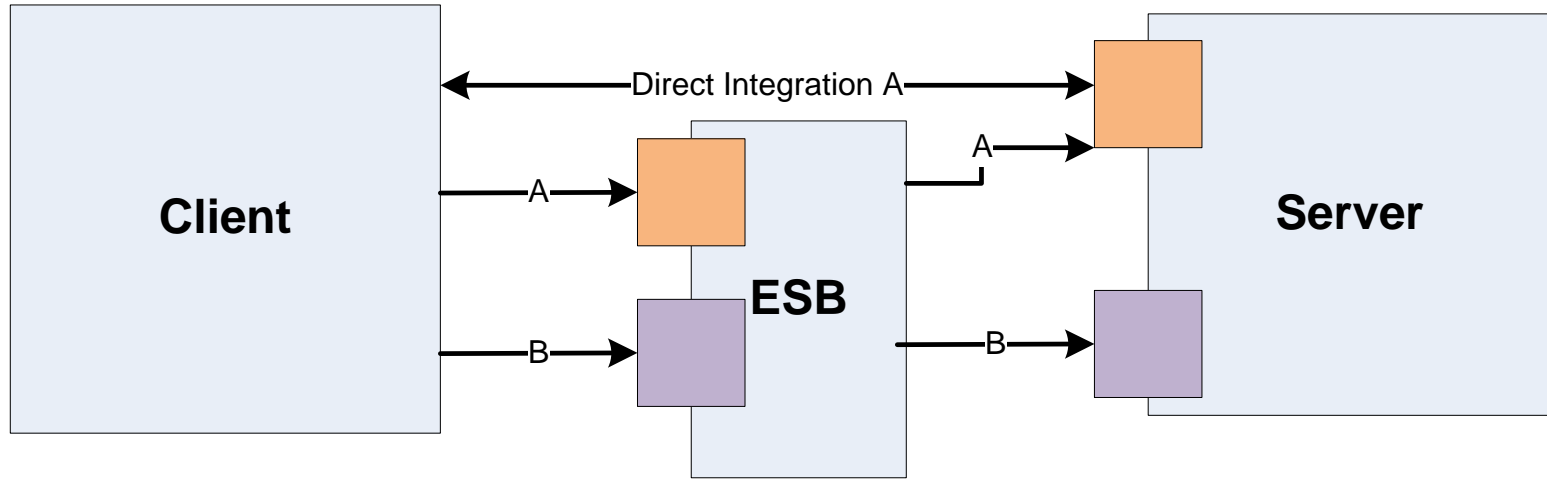


Generic-Typed: Message

- Example Request Message:

```
<ns2:RequestMessage xmlns="http://iec.ch/TC57/2011/schema/message"
  xmlns:ns3="http://iec.ch/TC57/2011/EndDeviceControls#">
  <Header>
    <Verb>create</Verb>
    <Noun>EndDeviceControls</Noun>
    <Revision>2.0</Revision>
    <Timestamp>2011-11-10T16:26:54.232Z</Timestamp>
    <Source>EIServer</Source>
    <ReplyAddress>http://somehost/CIM/CIMService</ReplyAddress>
    <MessageID>0d39be6c-ddb0-4056-b9e7-5f15d91658cc</MessageID>
    <CorrelationID>e3f37e15-10e6-43de-8b45-6537022f9065</CorrelationID>
  </Header>
  <Payload>
    <ns3:EndDeviceControls>
      <ns3:EndDeviceControl>
        <ns3:EndDeviceControlType ref="3.31.0.23"/>
        <ns3:EndDevices>
          <ns3:mRID>cf14ac6f-4829-4d32-bbc4-ed90ddfa2760</ns3:mRID>
        </ns3:EndDevices>
      </ns3:EndDeviceControl>
    </ns3:EndDeviceControls>
  </Payload>
</RequestMessage>
```

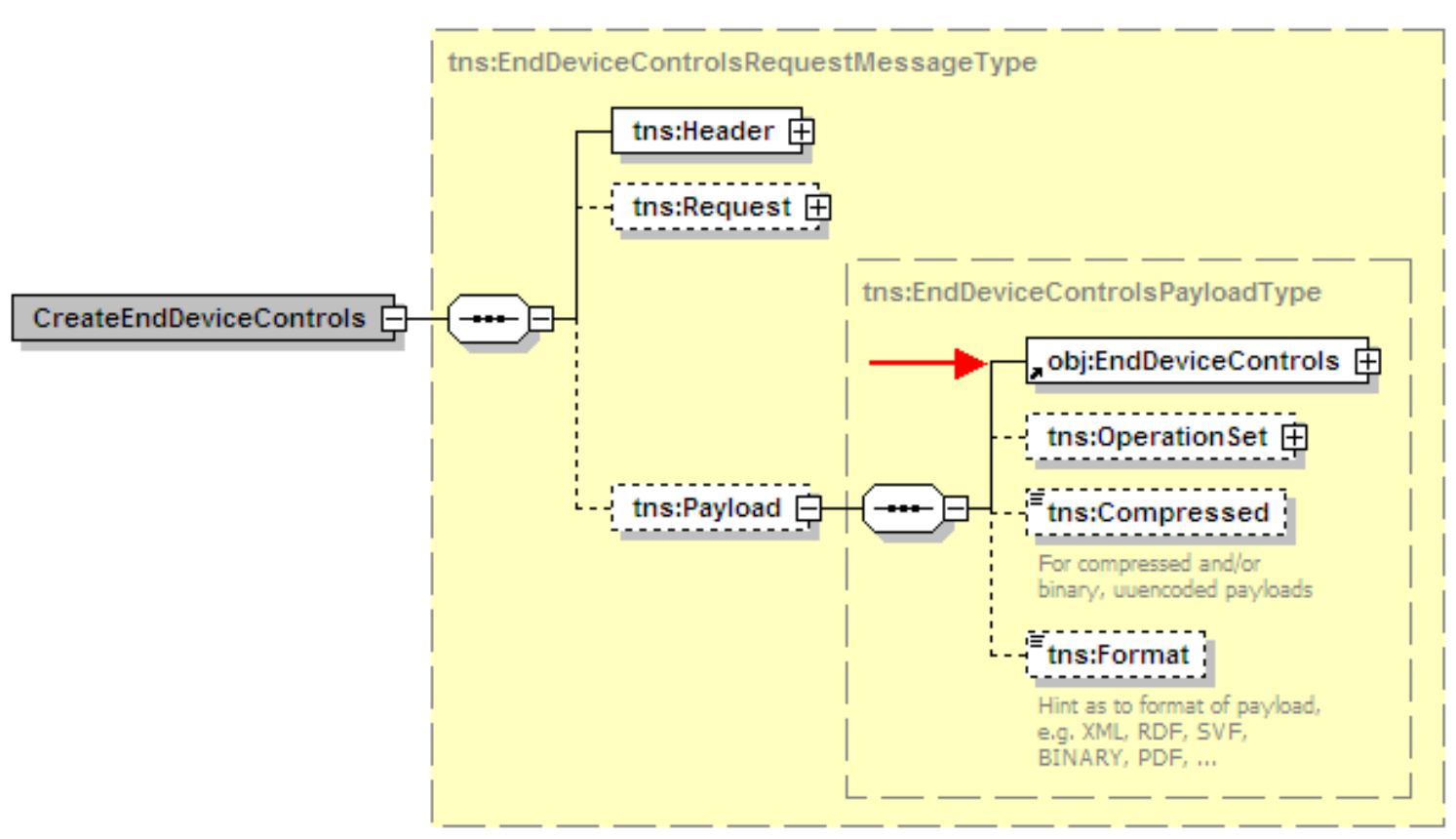
Strongly-Typed Web Services



- Uses multiple WSDLs to define messages corresponding to the Message Stereotypes based on CIM profile semantics.
- Payload is specific to one CIM profile type.
- Multiple endpoints to handle and route requests.
- Message routing can be handled via endpoint semantics without content introspection.
- Full message validation can be handled by the endpoint.



Strongly-Typed: Web Services



- `xs:any` replaced with specific CIM profile type.
- Allows full validation of message payload from WSDL definition.



Strongly-Typed: Services

- SOA model using CIM profile semantic based services.
- Service Types:
 - Client-based services
 - **Receive**: Consumes business objects from an external source.
 - Server-based services
 - **Execute**: Executes a service execution request.
 - **Get**: Executes a query request.
 - Intermediary (ESB) based services:
 - **Get**: Routes a query request.
 - **Show**: Provides a business object for consumption when state is not changed. (Response to a Get request.)
 - **Send**: Provides a business object for consumption when state has changed. (Unsolicited publication service.)
 - **Request**: Routes a service execution request.
 - **Reply**: Routes a response to an execute service request.
- Common interaction patterns will illustrate the usage.



Strongly-Typed: Operations

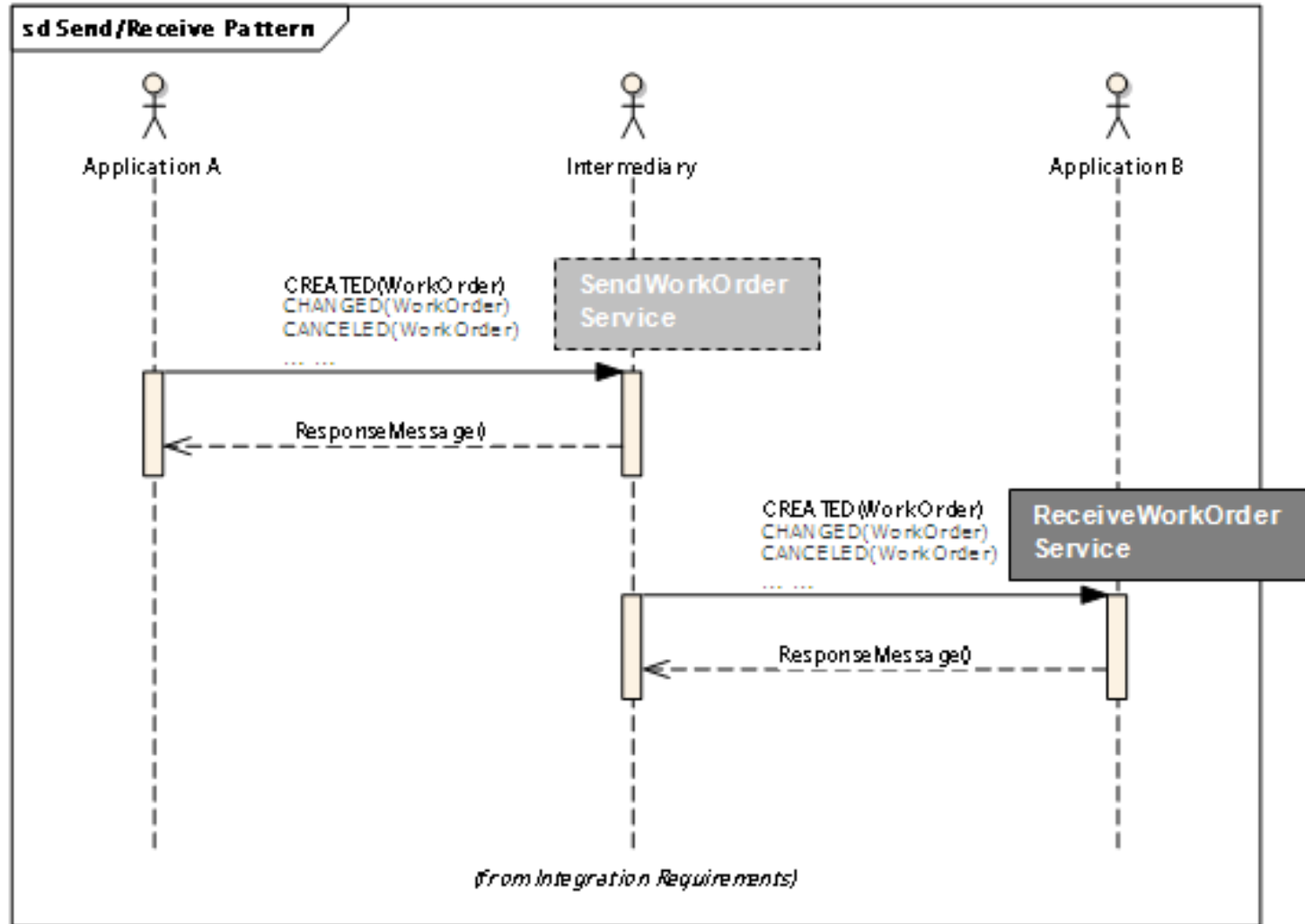
- Operations are based on 61968 Verbs:
 - Request and Execute Services:
 - **Create**: used to create objects of the type specified by the noun.
 - **Change**: used to modify objects.
 - **Cancel**: implies actions related to business processes, such as the cancellation of a control request.
 - **Close**: implies actions related to business processes, such as the closure of a work order.
 - **Delete**: used to delete objects.
 - Send, Reply and Receive Services:
 - **Created**: Event denoting creation of object(s).
 - **Changed**: Event denoting modification of objects(s).
 - **Canceled**: Event denoting cancellation of object(s).
 - **Closed**: Event denoting closure of object(s).
 - **Deleted**: Event denoting deletion of objects(s).
 - Get Services:
 - **Get**: used to query for objects of the type specified by the message noun.



Strongly-Typed: Naming Patterns

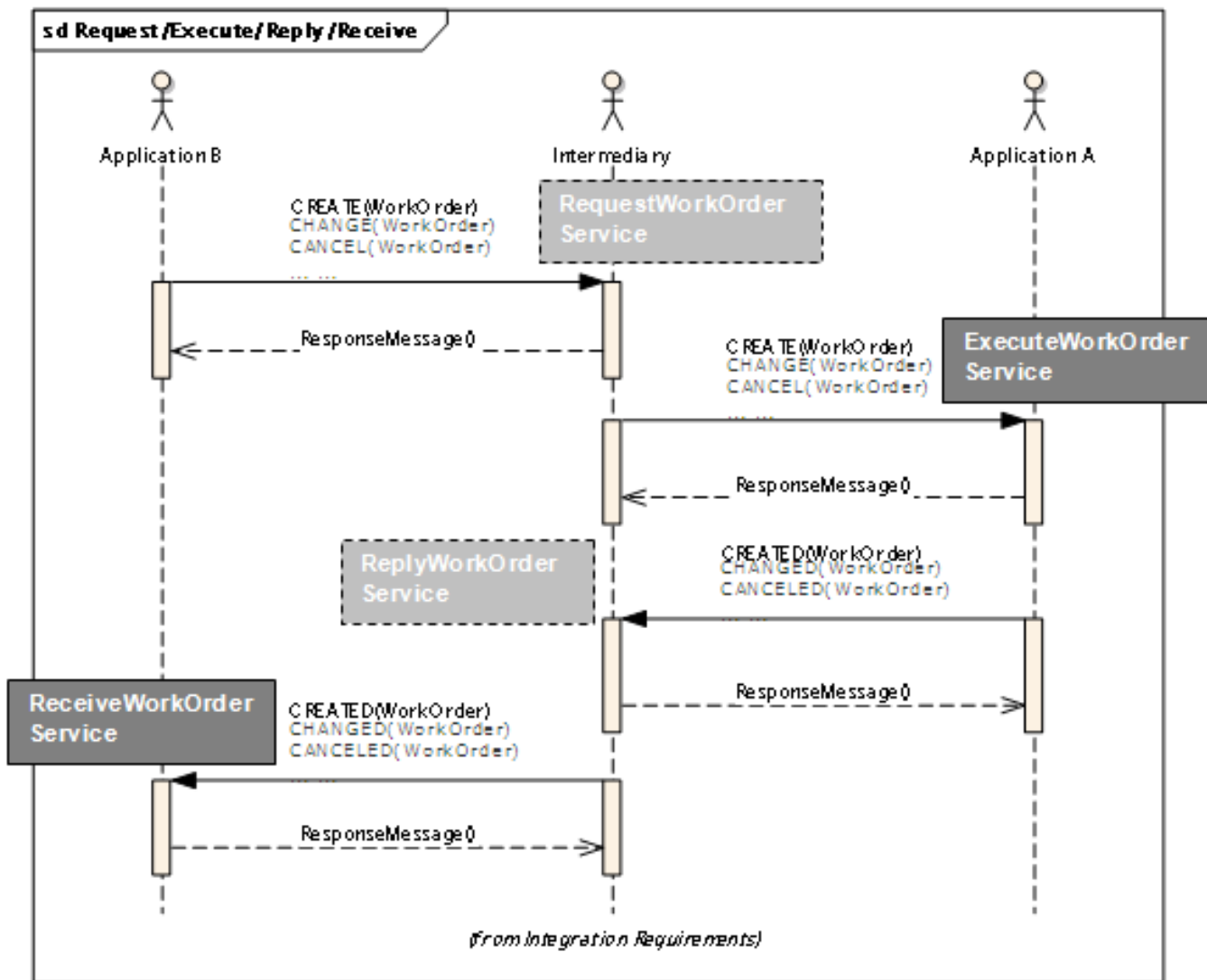
- Service Naming Pattern: <ServiceType><CIMProfile>
- Operation Naming Pattern: <Verb><CIMProfile>
- WSDLs are derived from Templates in 61968-100 Annex.
- Examples:
 - ExecuteEndDeviceControls Service:
 - CreateEndDeviceControls
 - ChangeEndDeviceControls
 - CancelEndDeviceControls
 - CloseEndDeviceControls
 - DeleteEndDeviceControls
 - SendEndDeviceEvent Service:
 - CreatedEndDeviceEvent
 - ChangedEndDeviceEvent
 - CanceledEndDeviceEvent
 - ClosedEndDeviceEvent
 - DeletedEndDeviceEvent

Strongly-Typed: Send/Receive





Strongly-Typed: Execute/Receive





Strongly-Typed: Message

- Example CreateEndDeviceControls message implemented by ExecuteEndDeviceControls service:

```
<ns2:CreateEndDeviceControls xmlns:ns2="http://iec.ch/TC57/2011/EndDeviceControlsMessage"
  xmlns="http://iec.ch/TC57/2011/schema/message"
  xmlns:ns3="http://iec.ch/TC57/2011/EndDeviceControls#">
  <ns2:Header>
    <Verb>create</Verb>
    <Noun>EndDeviceControls</Noun>
    <Revision>2.0</Revision>
    <Timestamp>2011-11-10T16:26:54.232Z</Timestamp>
    <Source>EIServer</Source>
    <ReplyAddress>http://somehost/CIM/ReceiveEndDeviceControls</ReplyAddress>
    <MessageID>0d39be6c-ddb0-4056-b9e7-5f15d91658cc</MessageID>
    <CorrelationID>e3f37e15-10e6-43de-8b45-6537022f9065</CorrelationID>
  </ns2:Header>
  <ns2:Payload>
    <ns3:EndDeviceControls>
      <ns3:EndDeviceControl>
        <ns3:EndDeviceControlType ref="3.31.0.23"/>
        <ns3:EndDevices>
          <ns3:mRID>cf14ac6f-4829-4d32-bbc4-ed90ddfa2760</ns3:mRID>
        </ns3:EndDevices>
      </ns3:EndDeviceControl>
    </ns3:EndDeviceControls>
  </ns2:Payload>
</ns2:CreateEndDeviceControls>
```

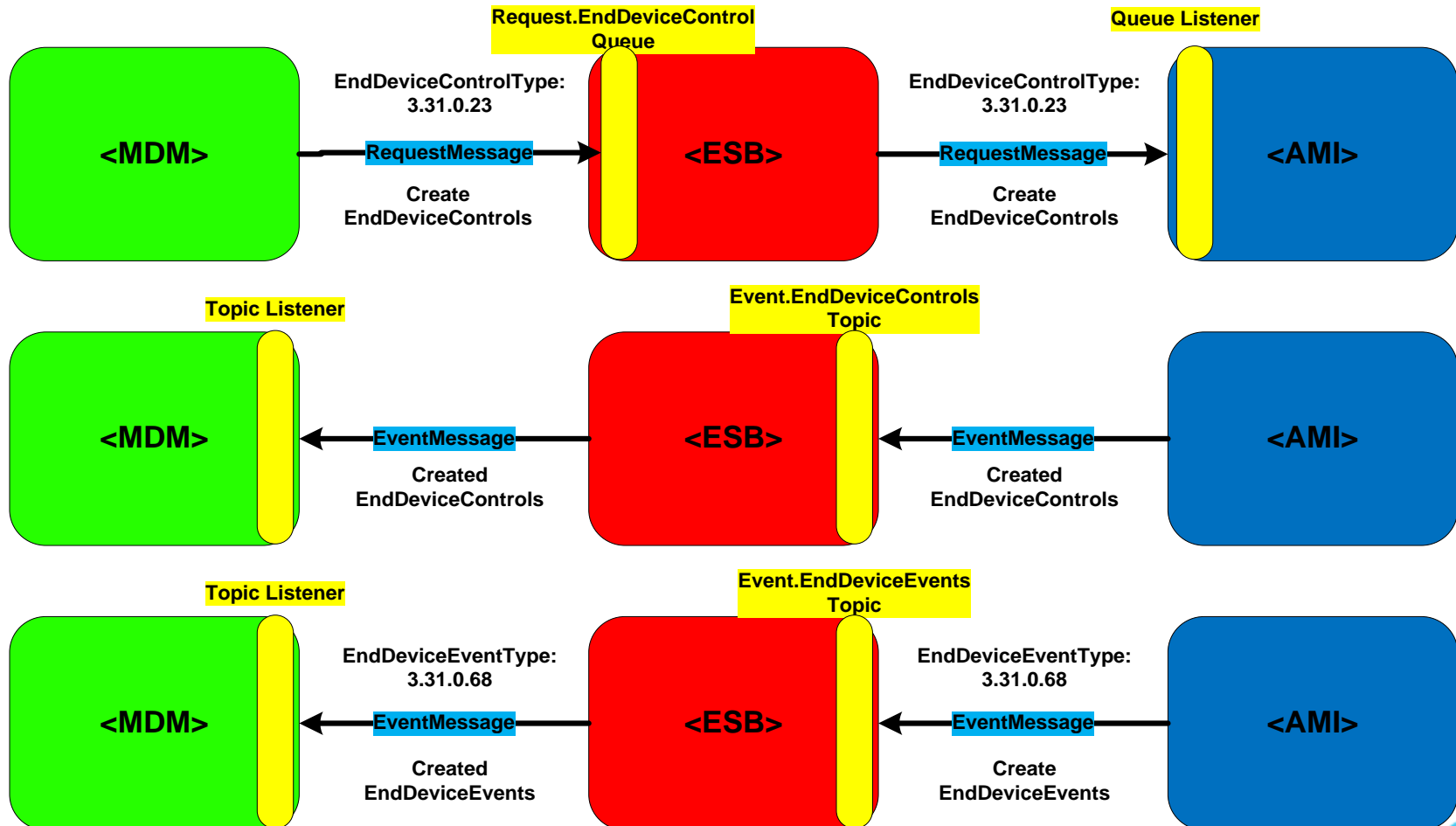


Interoperability Patterns and Use Cases

How can the three flavors achieve
interoperability in the enterprise?

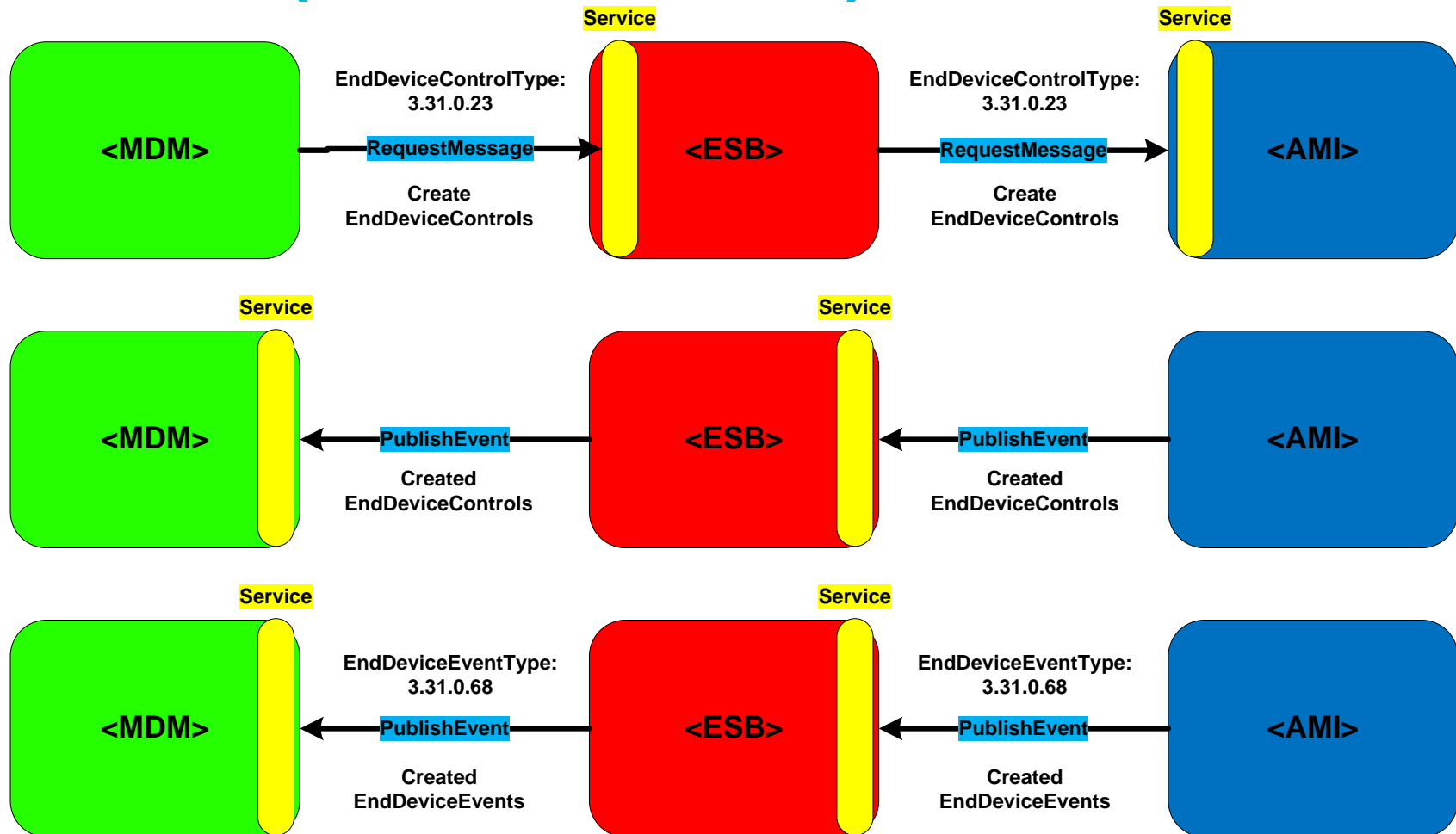


Use Case: MDM Disconnects AMI Meter (JMS)



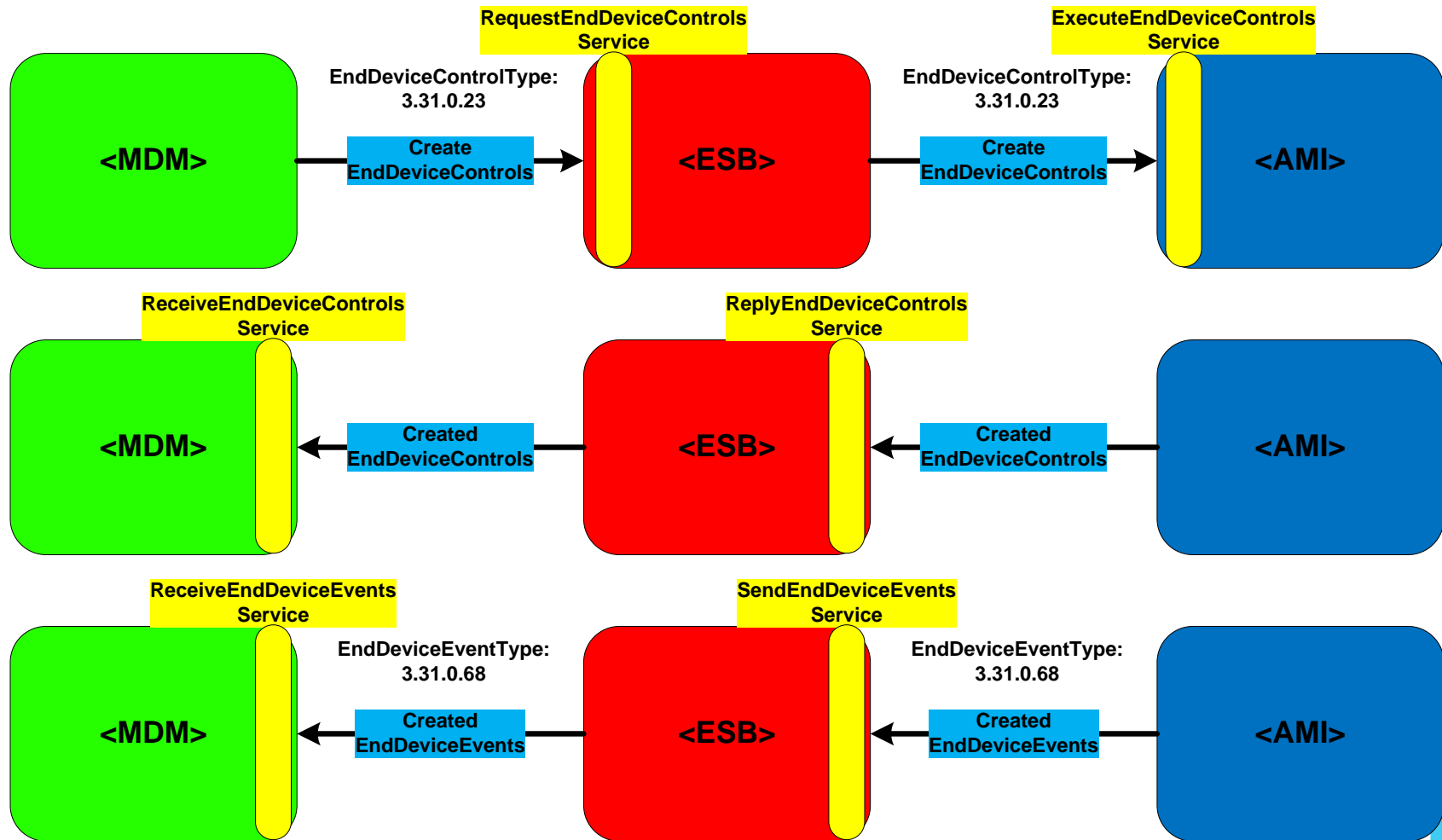


Use Case: MDM Disconnects AMI Meter (Generic WSDL)



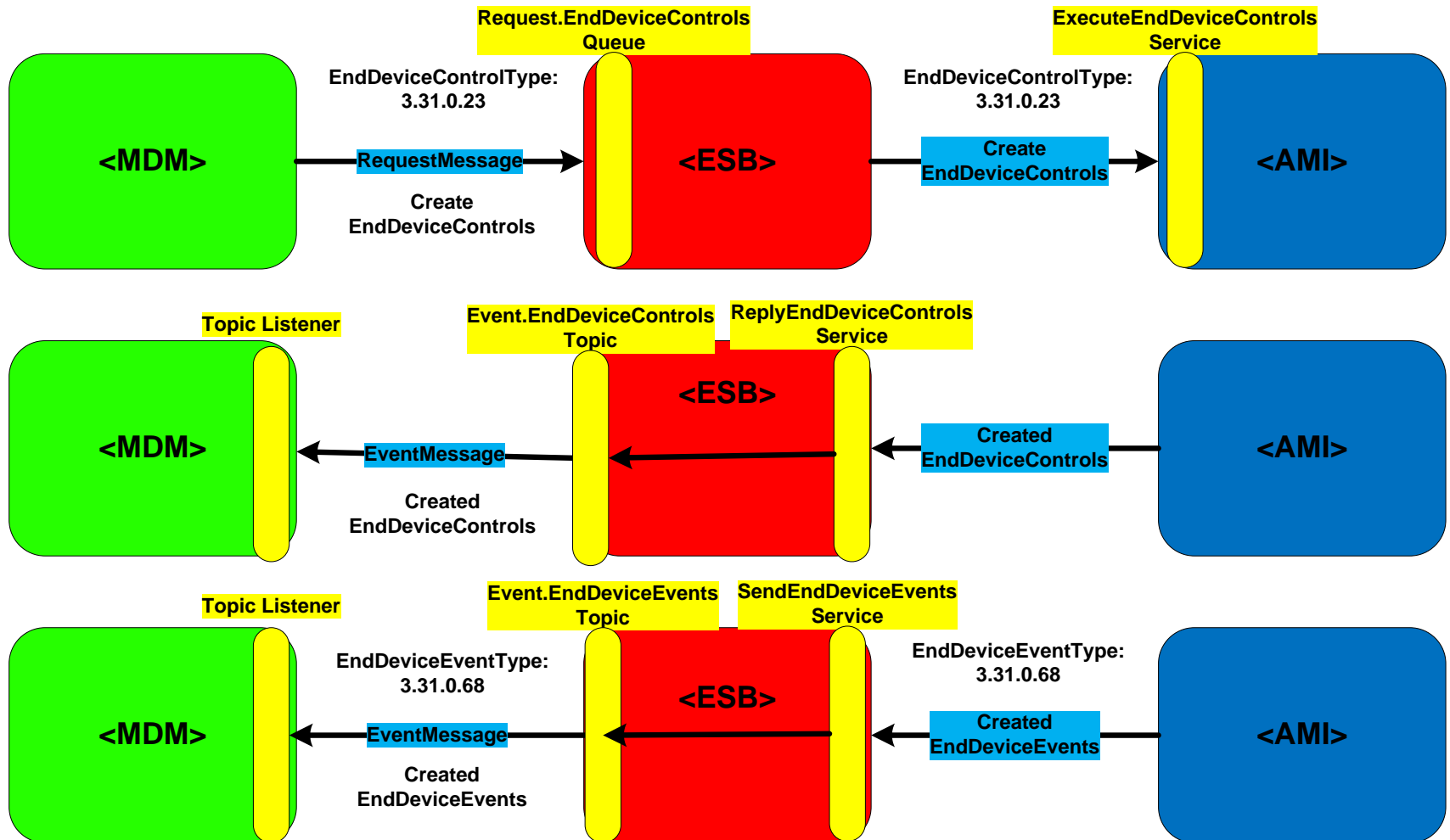


Use Case: MDM Disconnects AMI Meter (Strongly-Typed WSDL)





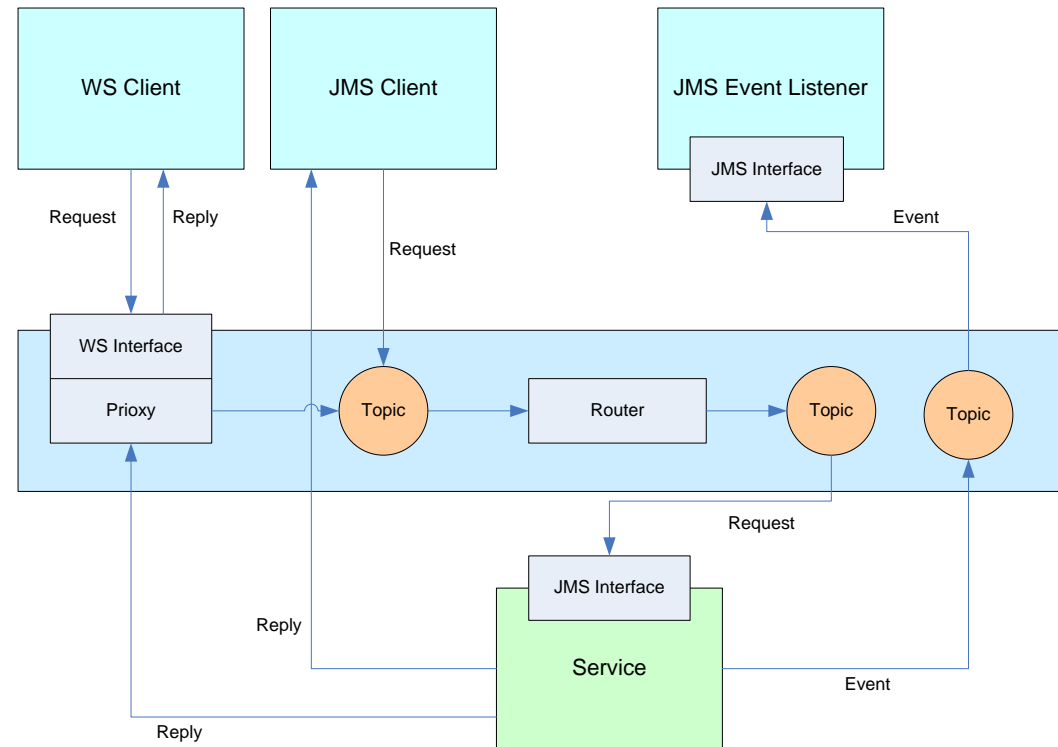
JMS and Web Service Interoperability





JMS Web Service Integration

- JMS-Centric Pattern
 - ESB hosts destinations
- Smart Router component does content-based routing.
- Smart Proxy implements web service interfaces and routes to appropriate destination.
- Replies and Events sent to appropriate destination/endpoint.





Other Considerations

Security, WSDL Generation, etc.

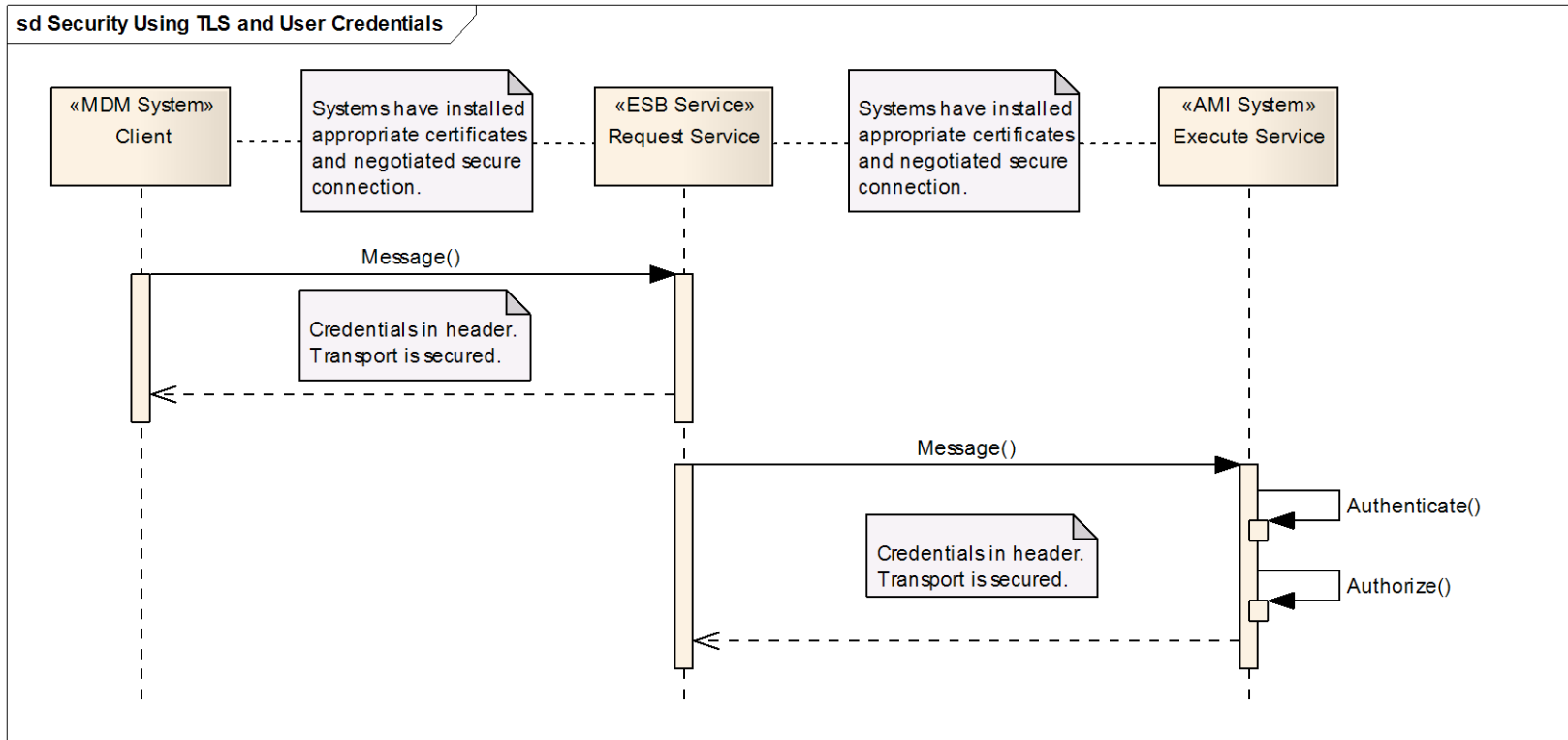


Security

- Goals
 - Reliably establish the identity of the entity interacting with the system. (Authentication)
 - Ensure that the entity is properly authorized to interact with the system in the manner requested. (Authorization)
 - Protect messages from eavesdropping and tampering.
- No specific implementation can be mandated.
 - Too many different approaches/requirements.
- A set of common strategies with recommended approaches:
 - Use of embedded credentials with secured transport. (SSL/TLS)
 - WS-Security



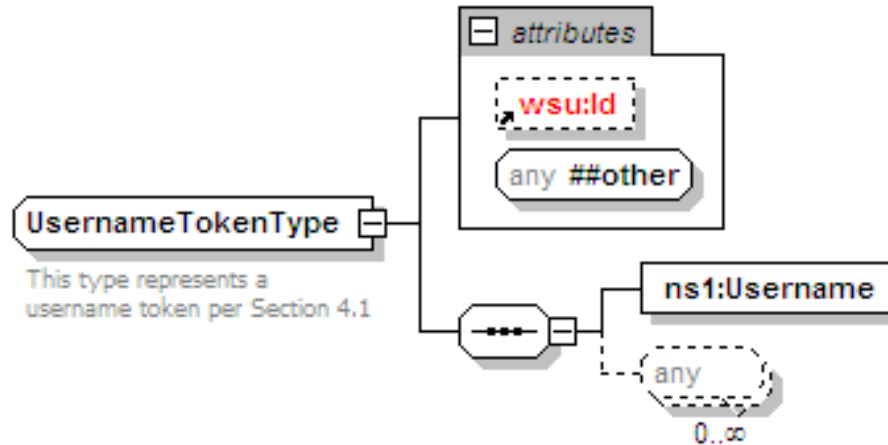
SSL/TLS Security



- Credentials (Username and Password) are in SOAP or JMS Header.
- Transport between systems encrypted.
- Details of how systems/certificates setup is outside scope.



User Credentials



- User credential schema from Web services Security UsernameToken Profile 1.1 OASIS standard specification.
- Leveraged to provide possible combined schema for user credentials across multiple strategies.



User Credential Recommendations

- Rather than using a cleartext password, it is recommended that a password digest be created and used in a Password field. This password should have a Type attribute of "wsse:PasswordDigest" and should be generated using the following formula:
$$\text{Password-digest} = \text{BASE64}(\text{SHA-256}(\langle \text{Nonce} \rangle + \langle \text{CreatedTimestamp} \rangle + \langle \text{Password} \rangle))$$

where:
 - $\langle \text{Nonce} \rangle$ is a random 16 byte value which is BASE64 encoded,
 - $\langle \text{CreatedTimestamp} \rangle$ is the ISO 8601 compliant creation timestamp,
 - $\langle \text{Password} \rangle$ is the secret password associated with the user account.
- Some applications in an effort to prevent exposure of the secret password to hacking attempts do not directly store the password, but rather store a hashed value and authenticate via application of the hashing algorithm and comparison with the expected stored hash value. In this case, the $\langle \text{Password} \rangle$ field would be the expected hashed value, rather than the password itself.
- The Nonce and CreatedTimestamp are optional in the UsernameToken element, but it is recommended that they be included in the UsernameToken to prevent replay attacks on the receiving system.



User Credential XML

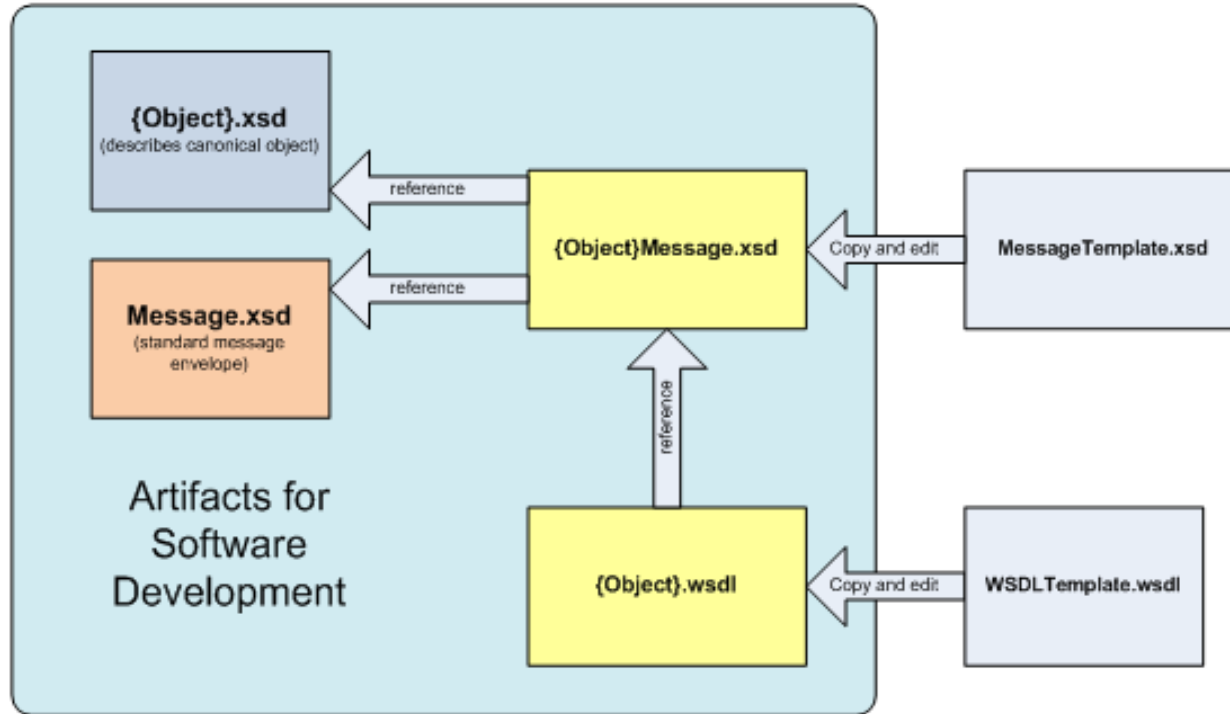
```
<wsse:UsernameToken>  
  <wsse:Username>scott</wsse:Username>  
  <wsse:Password Type="wsse:PasswordDigest">KE6QugOpkPyT3Eo0SEgT30W4Keg=  
  </wsse:Password>  
  <wsse:Nonce>5uW4ABku/m6/S5rnE+L7vg==</wsse:Nonce>  
  <wsu:Created xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">  
    2002-08-19T00:44:02Z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



Preventing Replay Attacks

- Replay attack prevention involves web service producers rejecting any message which does not have a Nonce and Created timestamp fields and applying appropriate verification processes on these fields.
- The Created timestamp allows web service producers to define a time window specifying how long a given message is to be considered valid. If the Created timestamp falls outside of the specified window the message is deemed stale and should be rejected. It is also recommended that web service producers keep a cache of nonces used within this freshness period and reject any messages that contain a previously used nonce. The combination of these two strategies will prevent replay attacks by limiting the time window of playback of messages and rejecting reuse of randomization parameters used in the generation of the password digest within that time window.

Strongly-Typed WSDL Generation



- Two sets of templates: MessageTemplate.xsd, WSDLTemplate.wsdl
- Copy each Template and perform string substitution.
- Produces Type-Specific Message.xsd and Service-Specific.wsdl.



Strongly-Typed Message Generation

```
<xs:schema xmlns:tns="http://iec.ch/TC57/2011/{Information_Object_Name}Message" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <!-- Base Message Definitions -->
  <xs:import namespace="http://iec.ch/TC57/2011/schema/message" schemaLocation="Message.xsd"/>
  <!-- CIM Information Object Definition -->
  <xs:import namespace="http://iec.ch/TC57/2011/{Information_Object_Name}#" schemaLocation="{Information_Object_Name}.xsd"/>
  <!-- PayloadType Definition -->
  <xs:complexType name="{Information_Object_Name}PayloadType">
    <xs:sequence>
      <xs:element ref="obj:{Information_Object_Name}"/>
      <xs:element name="OperationSet" type="msg:OperationSet" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

- Substitute EndDeviceControls for {Information_Object_Name} Yields:

```
<xs:schema xmlns:tns="http://iec.ch/TC57/2011/EndDeviceControlsMessage" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <!-- Base Message Definitions -->
  <xs:import namespace="http://iec.ch/TC57/2011/schema/message" schemaLocation="Message.xsd"/>
  <!-- CIM Information Object Definition -->
  <xs:import namespace="http://iec.ch/TC57/2011/EndDeviceControls#" schemaLocation="EndDeviceControls.xsd"/>
  <!-- PayloadType Definition -->
  <xs:complexType name="EndDeviceControlsPayloadType">
    <xs:sequence>
      <xs:element ref="obj:EndDeviceControls"/>
      <xs:element name="OperationSet" type="msg:OperationSet" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```



Strongly-Typed WSDL Generation

```
<wsdl:message name="Create{Information_Object_Name}RequestMessage">
  <wsdl:part name="Create{Information_Object_Name}RequestMessage" element="infoMessage:Create{Information_Object_Name}"/>
</wsdl:message>

<wsdl:message name="Change{Information_Object_Name}RequestMessage">
  <wsdl:part name="Change{Information_Object_Name}RequestMessage" element="infoMessage:Change{Information_Object_Name}"/>
</wsdl:message>

<wsdl:message name="Close{Information_Object_Name}RequestMessage">
  <wsdl:part name="Close{Information_Object_Name}RequestMessage" element="infoMessage:Close{Information_Object_Name}"/>
</wsdl:message>

<wsdl:message name="Cancel{Information_Object_Name}RequestMessage">
```

- Substitute EndDeviceControls for {Information_Object_Name} Yields:

```
<wsdl:message name="CreateEndDeviceControlsRequestMessage">
  <wsdl:part name="CreateEndDeviceControlsRequestMessage" element="infoMessage:CreateEndDeviceControls"/>
</wsdl:message>

<wsdl:message name="ChangeEndDeviceControlsRequestMessage">
  <wsdl:part name="ChangeEndDeviceControlsRequestMessage" element="infoMessage:ChangeEndDeviceControls"/>
</wsdl:message>

<wsdl:message name="CloseEndDeviceControlsRequestMessage">
  <wsdl:part name="CloseEndDeviceControlsRequestMessage" element="infoMessage:CloseEndDeviceControls"/>
</wsdl:message>

<wsdl:message name="CancelEndDeviceControlsRequestMessage">
```



Status

Where is the standard now?



Status

- Not a standard yet, but already being used in:
 - Interoperability tests
 - Vendor Products
 - Utility Projects
- Committee Draft out to National Committees for comments in the IEC.
- Fast tracked in IEC: Country Comments will be incorporated and another committee draft for voting will be issued shortly for country comments and voting.

Thank you

Michael Johnson

Program Manager – EnergyAxis Solutions

michael.s.johnson@us.elster.com



elster
Vital Connections