The Standards Based Integration Company

Systems Integration Specialists Company, Inc.

IEC 61968 – 100: Implementation Profile

CIM University
Prague, Czech Republic
May 10, 2011

Margaret Goodrich, Manager, Systems Engineering
SISCO, Inc.
6605 19½ Mile Road
Sterling Heights, MI 48314 USA
Tel: +1-903-477-7176
Fax: +1-903-489-0063
E-Mail: margaret@sisconet.com

Introduction

- Scope/Purpose
- Some Definitions
- Use Case Sequence Diagrams
- Message Exchange Patterns
- Integration Patterns
- Other Topics:
  - Message Organization
  - Interface Specifications
  - ESB Considerations
  - Security Considerations

Scope/Purpose

- To Define an Implementation Profile for integration using an ESB that supports both JMS and Web Services.
- This standard defines how message payloads are conveyed using Web Services and the Java Message Service (JMS)
- The Goal is to provide sufficient information to enable implementations to be interoperable.

Scope/Purpose

- Provides recommendations for Web Service implementation using Service Oriented Architecture (SOA) service patterns and Web Services Description Language (WSDL).

- Was used as an integral part of the interoperability tests for distribution management systems in 2009 and 2011.

Some Definitions

- Enterprise Service Bus (ESB) – refers to a software architecture construct that provides foundational services via an event-driven and standards-based messaging engine.
- Java Message Service (JMS) - is an API that supports request/reply, publish/subscribe, and point-to-point messaging patterns.
- Service Oriented Architecture (SOA) – is an architectural style for creating and using business processes, packaged as *services*.
- Simple Object Access Protocol (SOAP) – is a standard that defines the formatting of XML messages and serves as a foundation layer of the Web Services Protocol stack.

Some Definitions - Continued

- Web Services Definition Language (WSDL) – is an XML-based language that is used to describe Web Services. WSDL is used in combination with SOAP or XML schema to provide Web Services over the internet.
- XML Schema – is one of several XML schema languages. Provides a set of rules to which an XML document must conform in order to be considered "valid". An XML Schema instance is an XML Schema Definition (XSD).

Some Definitions - Continued

- JMS Elements:
  - JMS Provider – an implementation of the JMS interface for Message Oriented Middleware (MOM)
  - JMS Client – an application or process that produces and/or receives messages
  - JMS Producer – A JMS Client that creates and send messages
  - JMS Consumer – A JMS Client that receives messages
  - JMS Message – An object that contains the data being transferred between JMS clients.

Some Definitions - Continued

- JMS Elements - Continued:
  - JMS Queue – Staging area for message that have been sent and are waiting to be read. Messages are delivered in the order sent. A message is removed once it is read.
  - JMS Topic – A distribution mechanism for publishing messages that are delivered to multiple subscribers.

Some Definitions - Continued

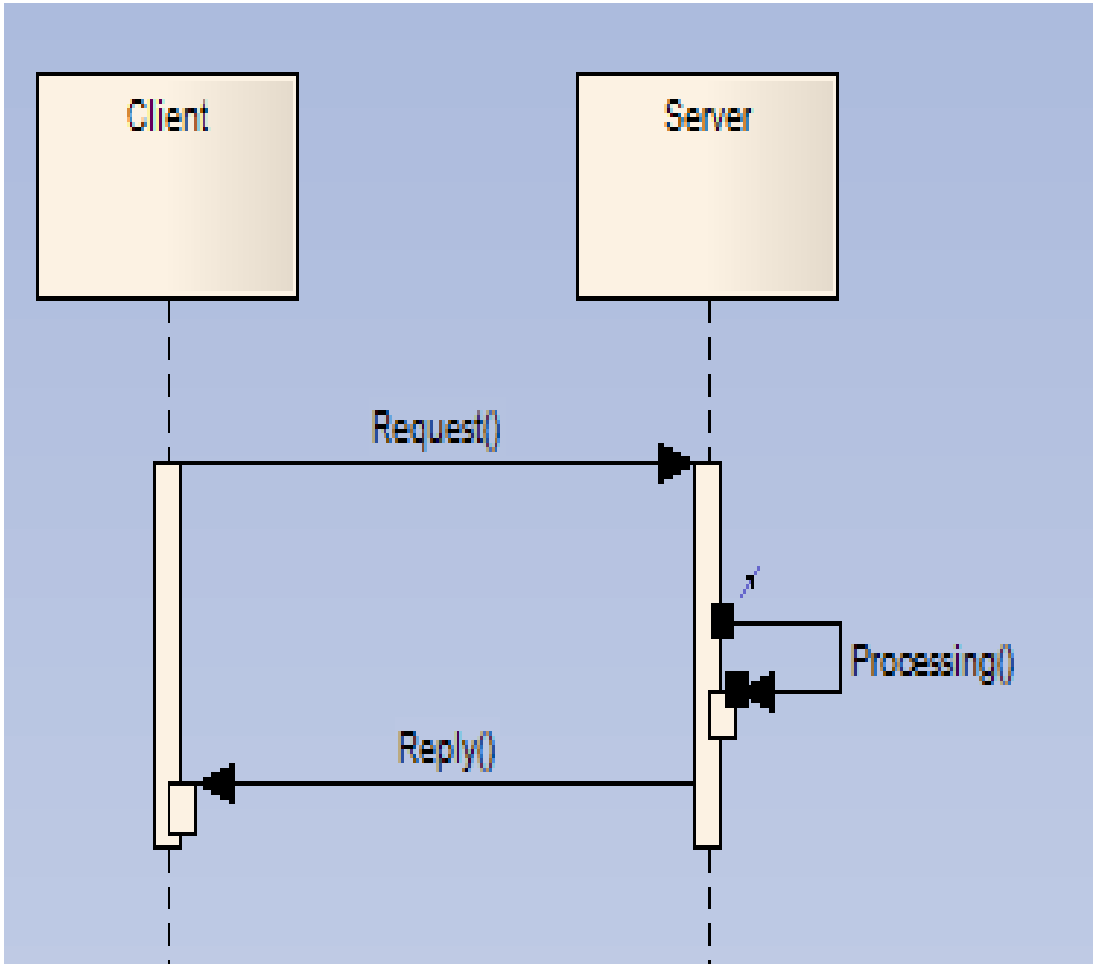
- Differences and Similarities between JMS Topics and JMS Queues:
  - Topics are used when the destination of a message is potentially more than one process
  - Queues are used when the destination of a message is at most one process
  - Topics and Queues are organized and named hierarchically
  - Except for the use of a durable subscription, a process can only receive a copy of a message published to a topic if it is running and has an active subscription
  - Message published to queues will remain on the queue until de-queued by a receiving process

Use Case Sequence Diagrams

- The document describes several use cases related to the interactions between components with a set of systems.
- Use Cases presented are from the perspective of the integration of systems; that is, the actors are defined in terms of the software systems that need to interact: (1) Client; (2) Server; (3) ESB, and; (4) Adapter
- Use Cases are described using Sequence Diagrams
- Three key terms related to messaging are request, reply and event and are reflected in terms of the verbs used to define the information flows.

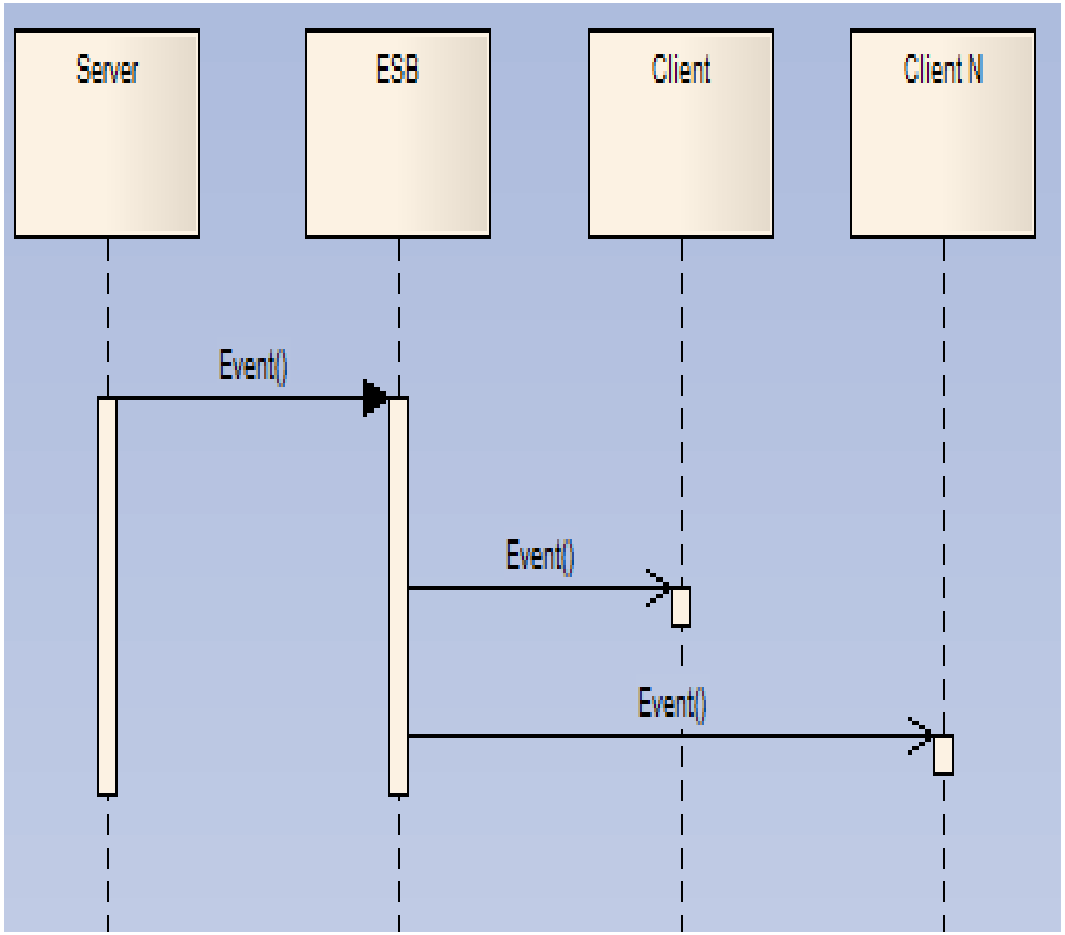
Sequence Diagrams – Simple Request/Reply

- Client makes query request & server returns a set of objects
- Client makes a transaction request & server creates or modifies a set of objects
- Request uses *get, create, update, delete, close or cancel* as the verbs.



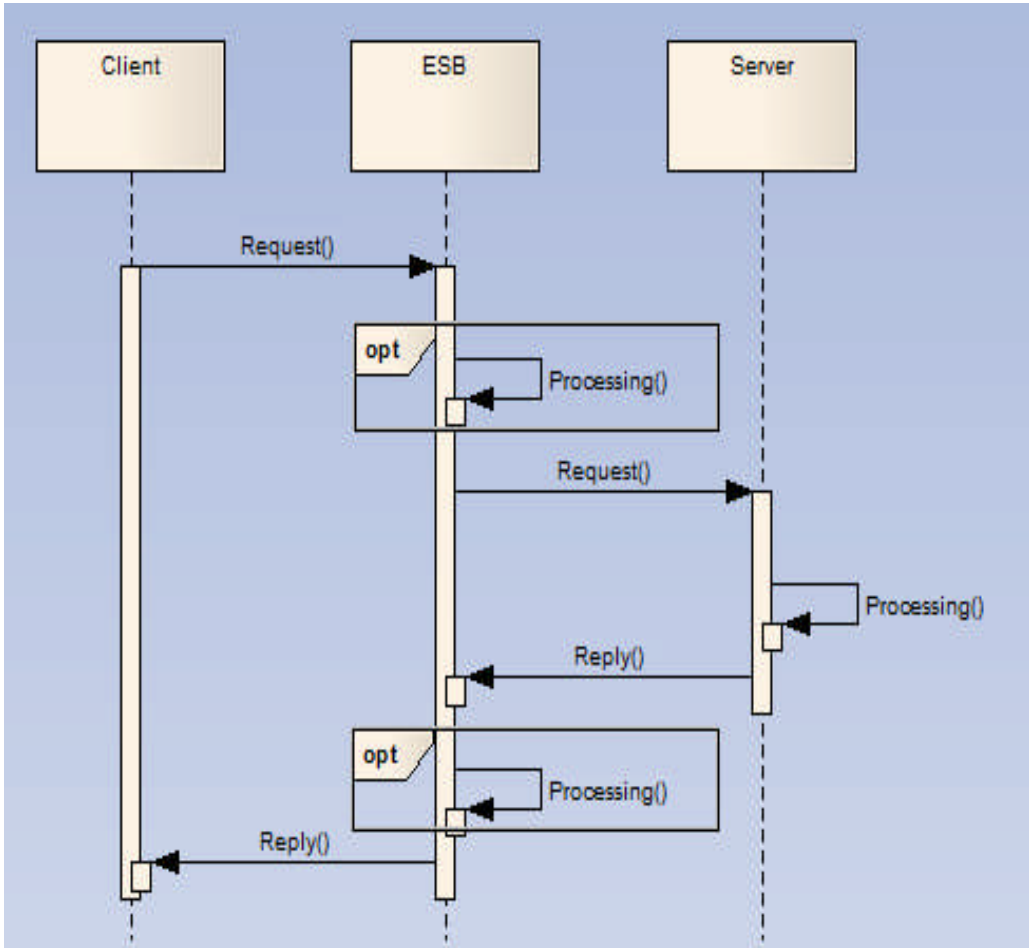
Sequence Diagrams – Request/Reply Using ESB

- Client processes can subscribe to “listen” for events
- Events use “past tense” verbs: *(1) created; (2) updated; (3) deleted; (4) cancelled, and; (5) closed*



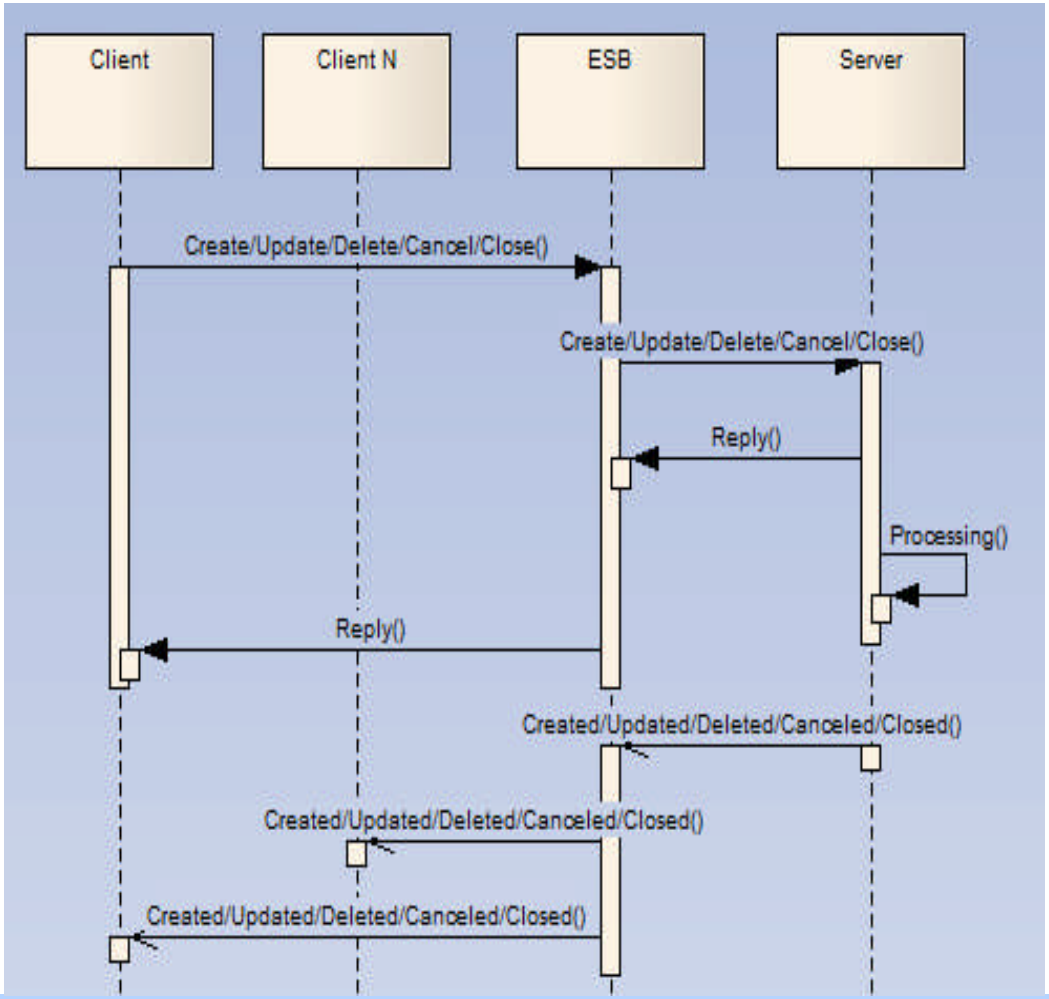
Sequence Diagrams – Events

- ESB decouples client from server
- Client does not have to know the location of the server or conform to the exact interface since the ESB completes the transformation
- Routing and mapping takes place in the integration layer



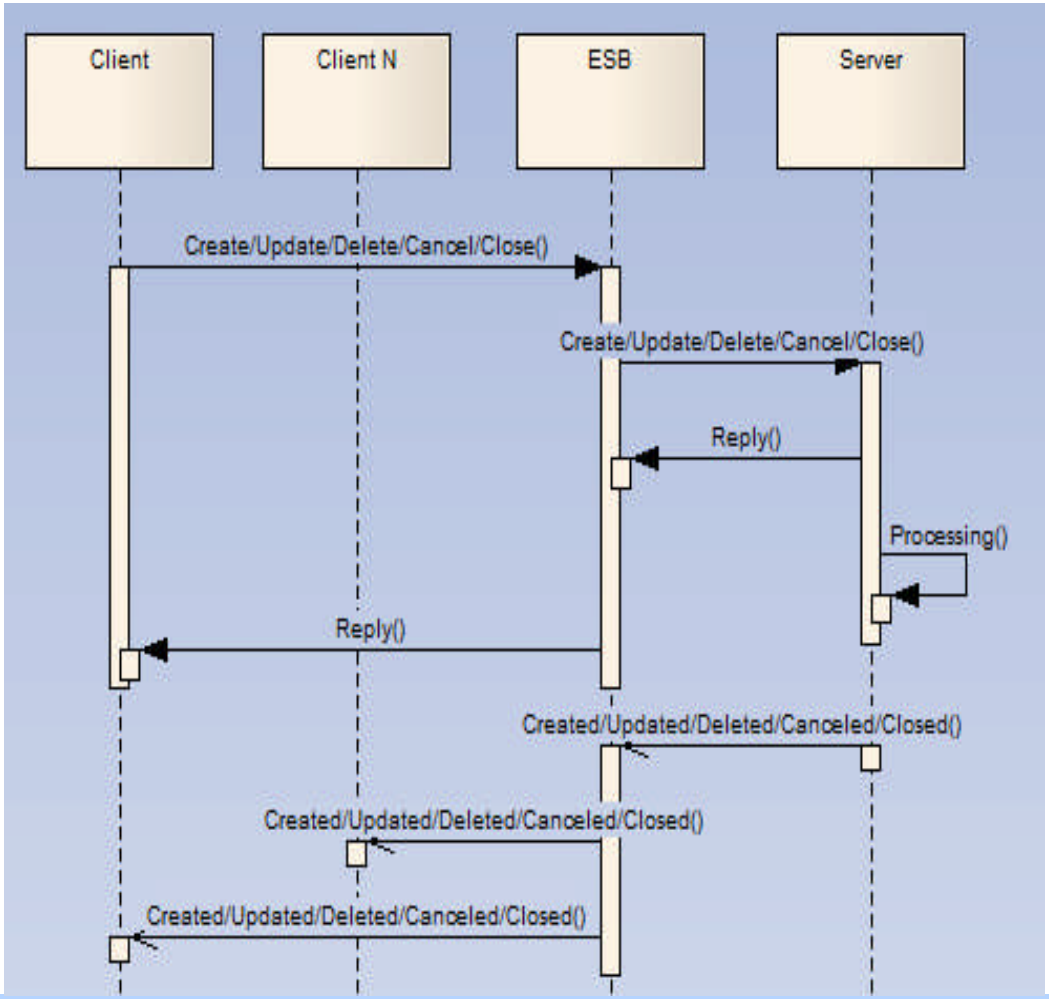
Sequence Diagrams – Transactions

- Typically a combination of a request/reply for a message exchange with a consequential publication of events
- Requests related to transactions use the verbs: *(1) create; (2) update; (3) delete; (4) cancel, and; (5) close*



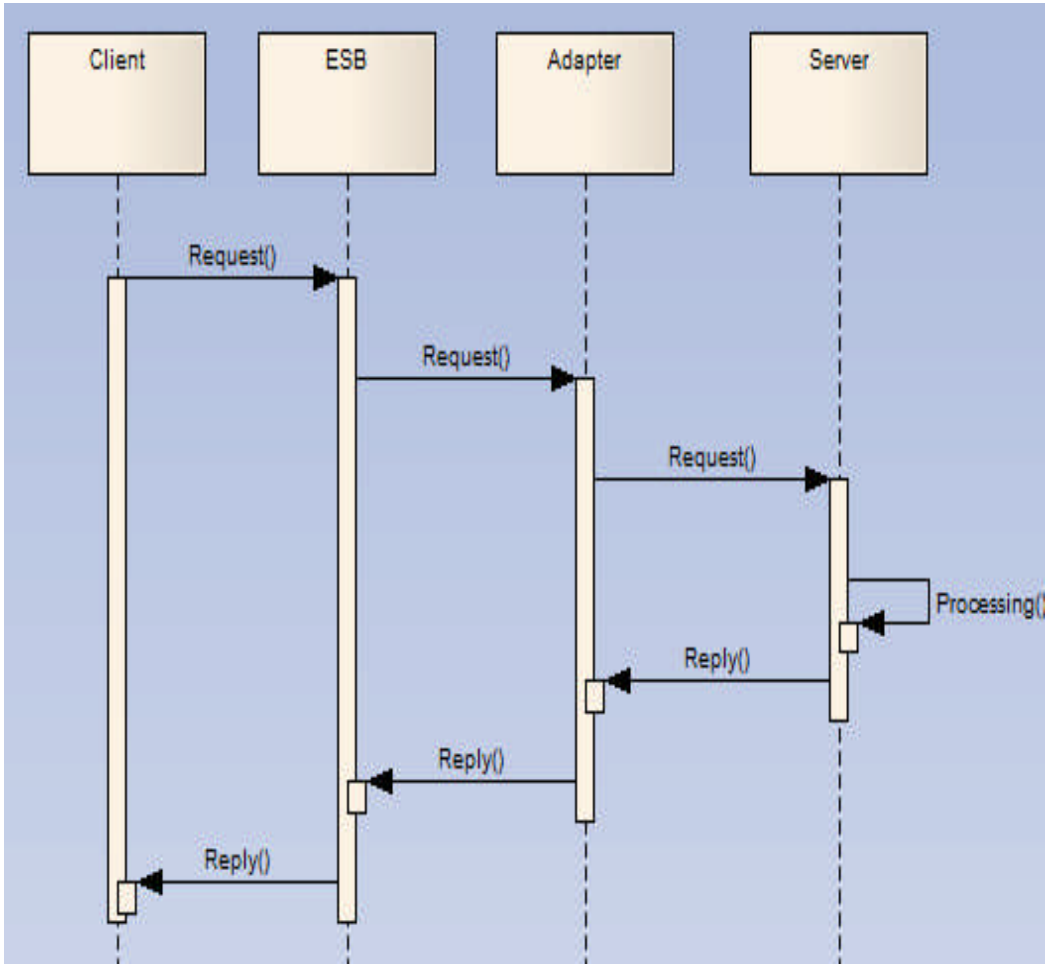
Sequence Diagrams – Callbacks - Update

- BLAH – BLAH Typically a combination of a request/reply for a message exchange with a consequential publication of events
- Requests related to transactions use the verbs: *(1) create; (2) update; (3) delete; (4) cancel, and; (5) close*



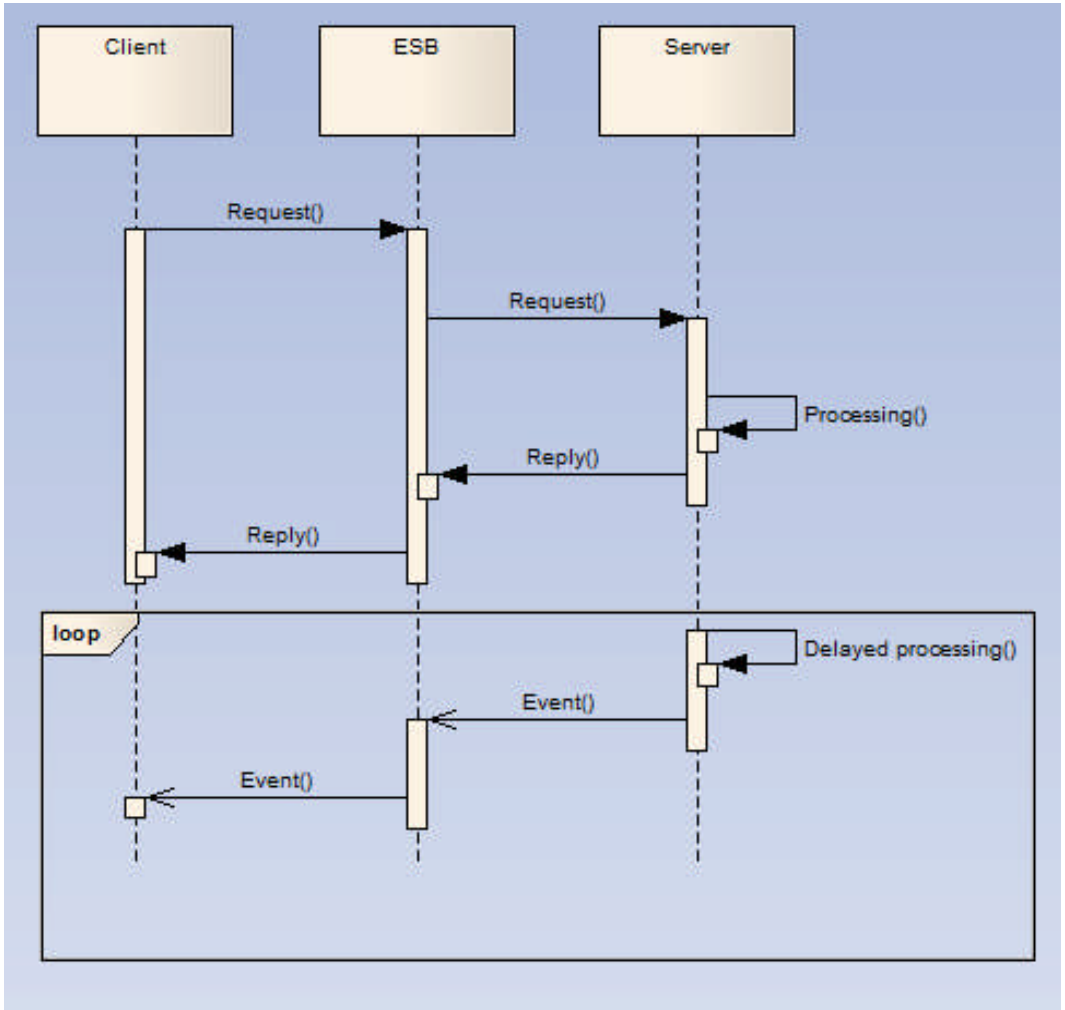
Sequence Diagrams – Adapters

- When a System cannot directly connect to an ESB, an adapter is used to handle the connection.
- The system may be a database or a file directory.
- Adapter can generate events on behalf of the system



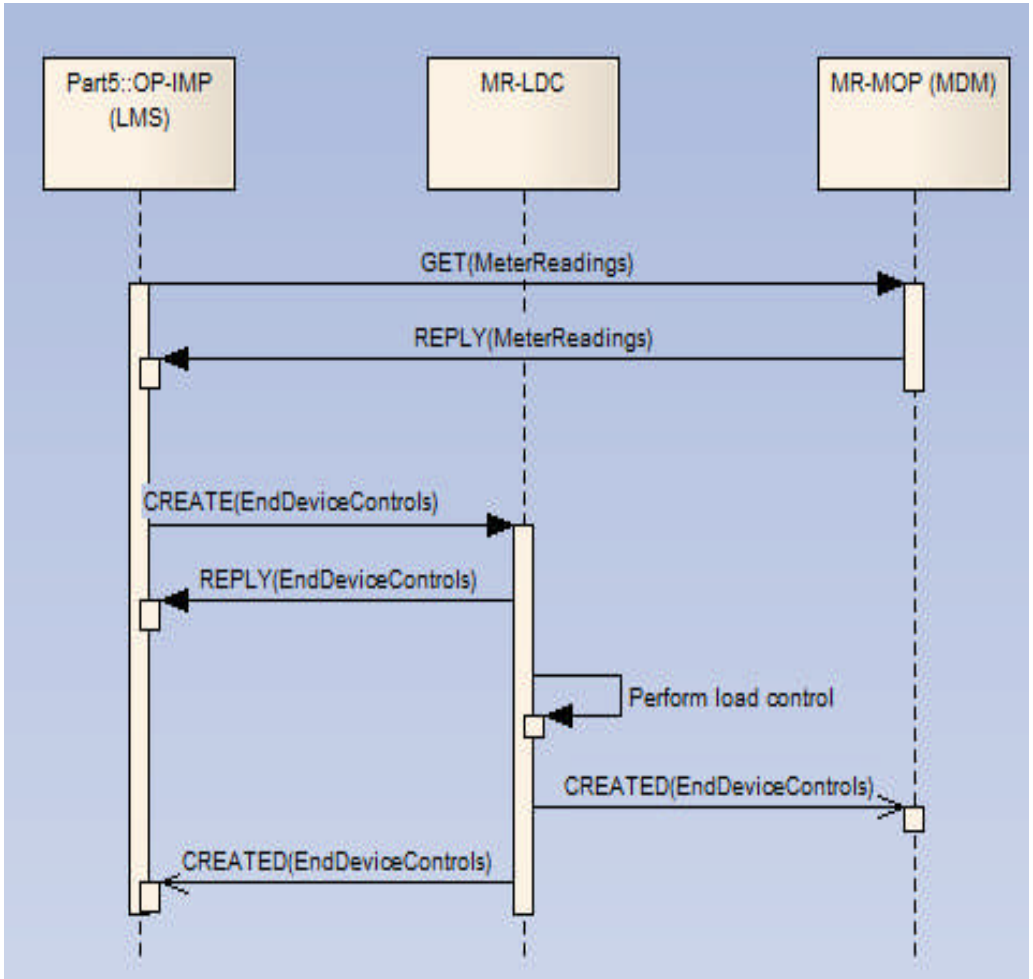
Sequence Diagrams – Complex Messaging

- Used when a request involves many asynchronous replies to return information as it becomes available.
- Results are returned in the form of asynchronous events.
- These messages may take significant time to obtain results.



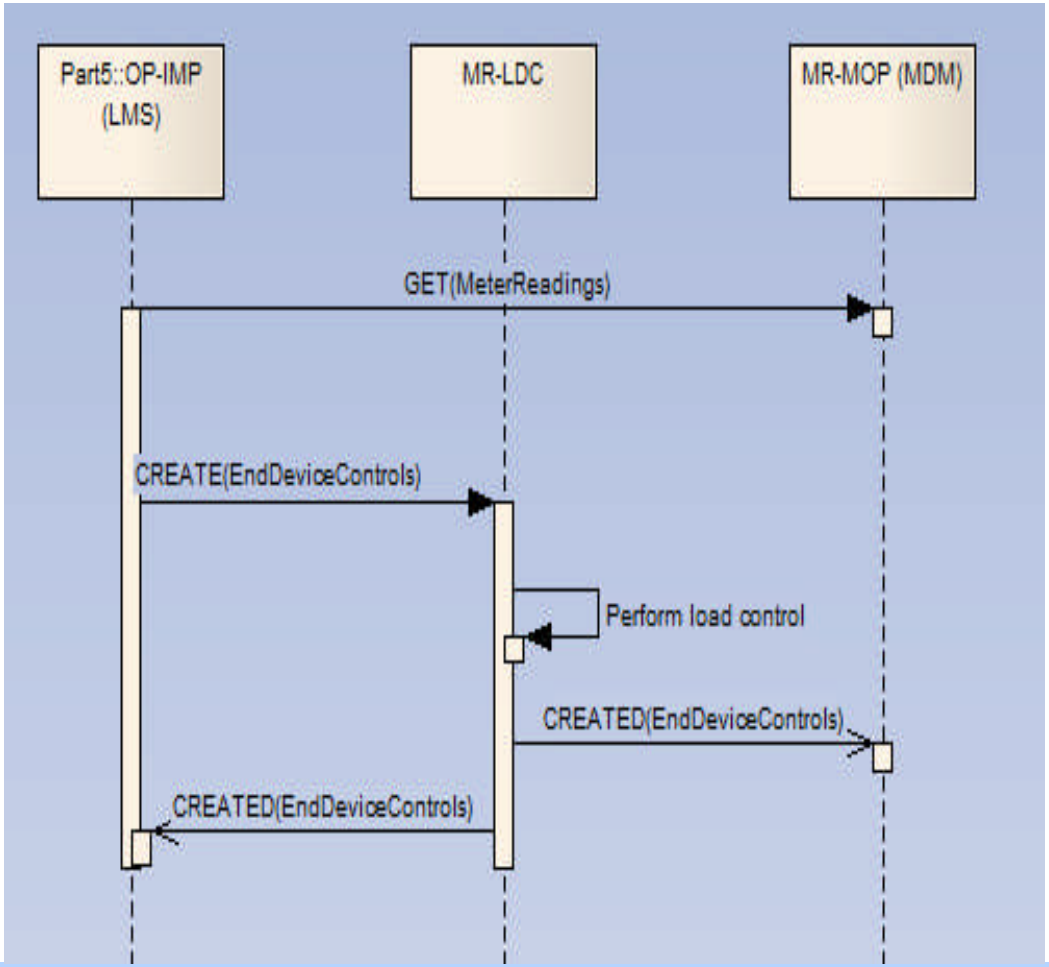
Sequence Diagrams – Application Level

- Application-level messages are defined in the form of *VERB(Noun)*
- Verbs are *Get, Reply, Create* and *Created*
- Nouns are *MeterReading* and *EndDeviceControls*



Sequence Diagrams – Application Level w/Synchronous Replies

- For Synchronous Request/Reply message, the REPLY messages are assumed to be immediately returned.

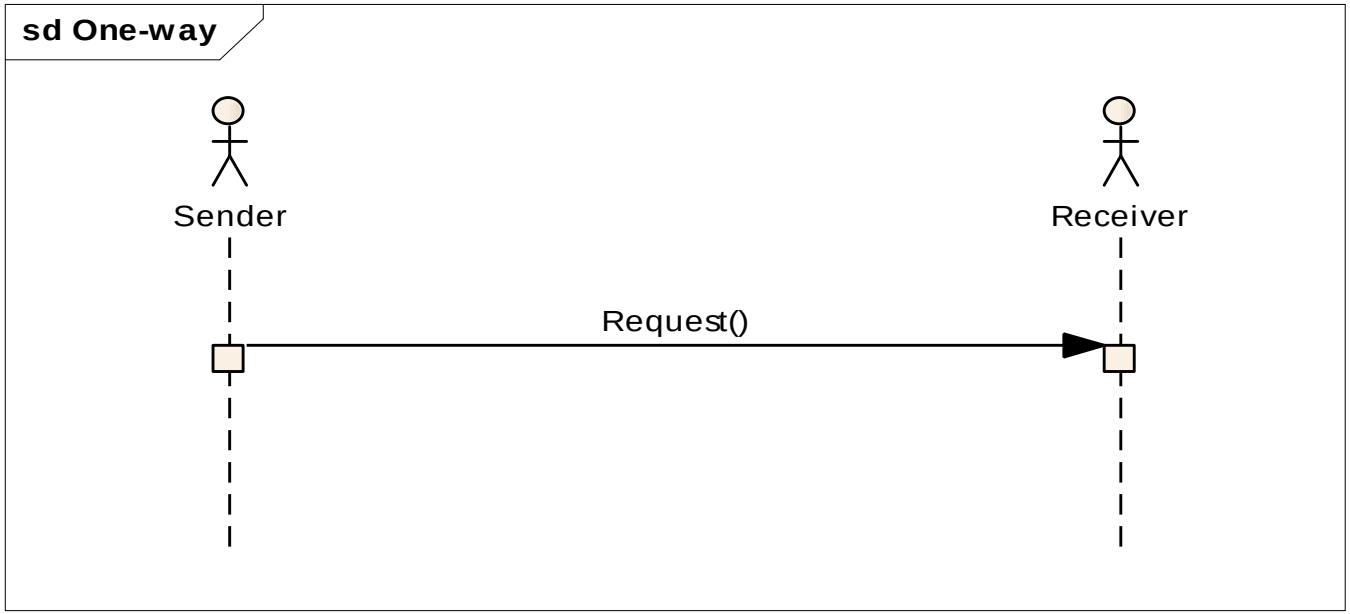


Message Exchange Patterns

- Three basic types:
  - One Way
  - Two Way
  - Call Back

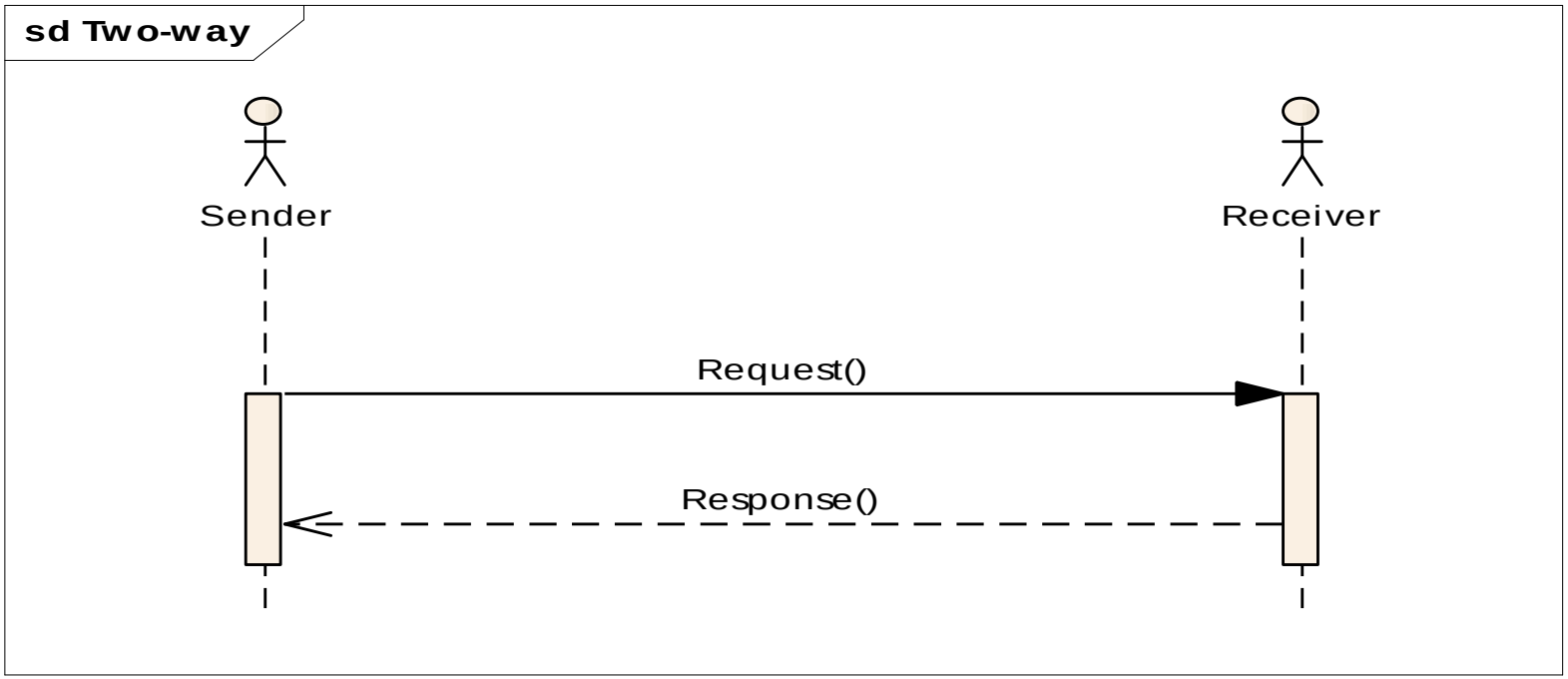
Message Exchange Pattern: One Way

- The sender sends its request out but does not expect a response message back.
- Since typically HTTP protocol is used, the sender can still get HTTP-level information such as a HTTP 404 error for a failed server communication.



Message Exchange Pattern: Two Way

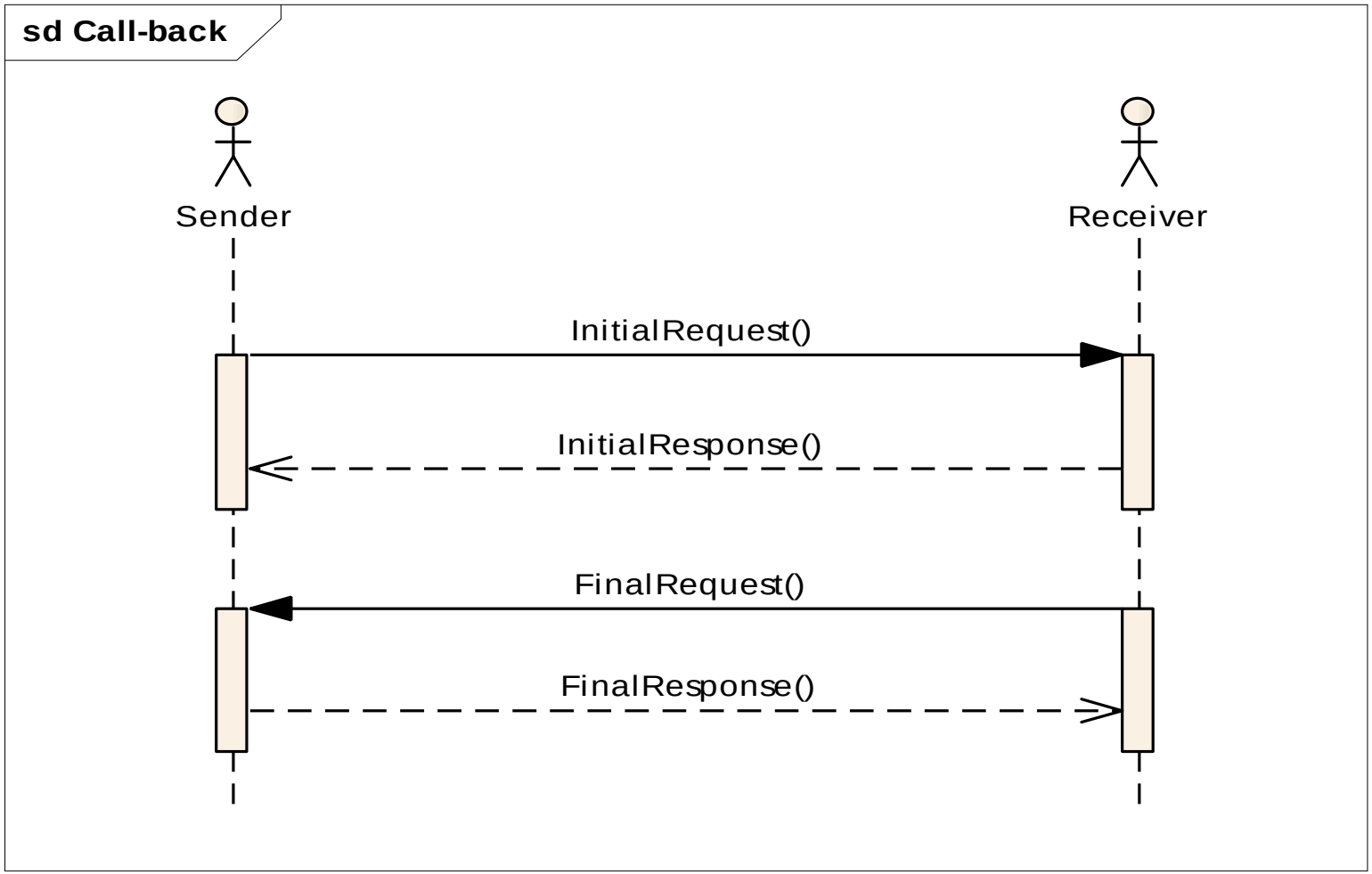
- Is a synchronous process with typically two messages involved, one for request and one for response.
- A sender sends a request to a receiver who returns a response message back to the sender. The request message is indicated as Request and the response message Response.



Message Exchange Pattern: Call Back

- The callback is an asynchronous process for message exchange.
- It is made of two request/response (initial and final) synchronous calls.
- The two are correlated in a way that each party can unambiguously identify which callback goes with which initial request.
- The Sender sends an initial request to Receiver with a message called InitialRequest.
- The Receiver receives the message, returns a response message back and invokes the final request/response call with a request message called FinalRequest.
- The whole call-back process is completed after the Sender replies the final request.

Message Exchange Pattern: Call Back



Integration Patterns

- There are two Patterns Basic Patterns:
  - Service Level Patterns – these include Send, Request, Reply, Retrieve, Receive, Execute or Show
  - Operation Level Patterns – these include Create, Change, Cancel, Close, Delete, Created, Changed, Canceled, Closed, Deleted, Get

Integration Patterns

- Using the two patterns, a service/operation naming convention for web services is described as:
  - **Service name**:
    - To follow ***<Service pattern name>+<Information Object>*** such as ExecuteEndDeviceControl
  - **Operation name**:
    - To follow ***<Operation pattern name>+<Information Object>*** such as CreatedEndDeviceControl

Integration Patterns

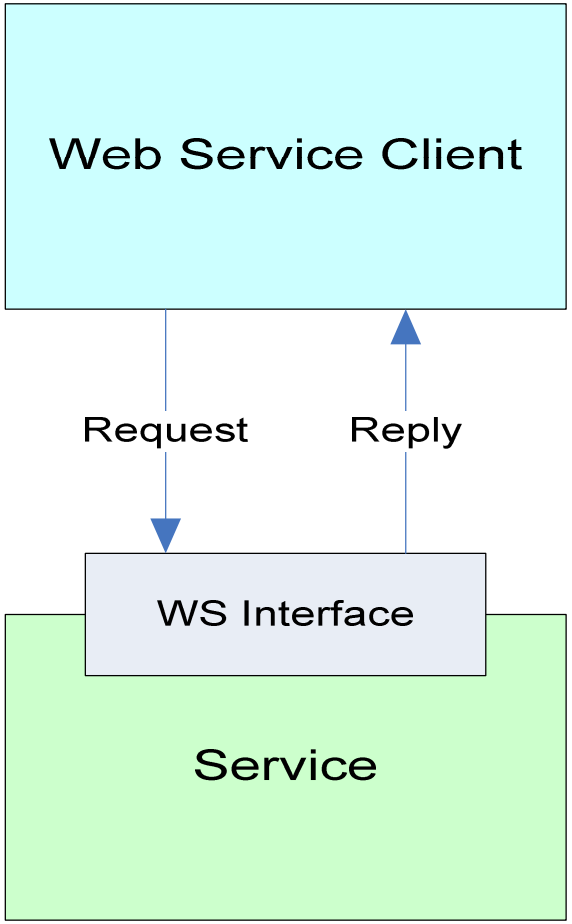
- Several integration patterns are described:
  - Synchronous request/reply using web services
  - One way requests using web services (source to target without reply)
  - Synchronous request/reply (source to target) with call backs (target to source)
  - Synchronous request/reply using JMS topics
  - Synchronous request/reply using JMS queues
  - Asynchronous request/reply using JMS topics
  - Asynchronous request/reply using JMS queues

Integration Patterns

- Integration patterns (Continued):
  - Publish/subscribe, where potentially many targets are listening for JMS messages on a topic (these messages are often referred to as events or notifications)
  - Point to point, where the target is listening for JMS messages on a queue
  - Publish/subscribe, where the client is listening at a specified URL for events that are to be routed by an ESB intermediary

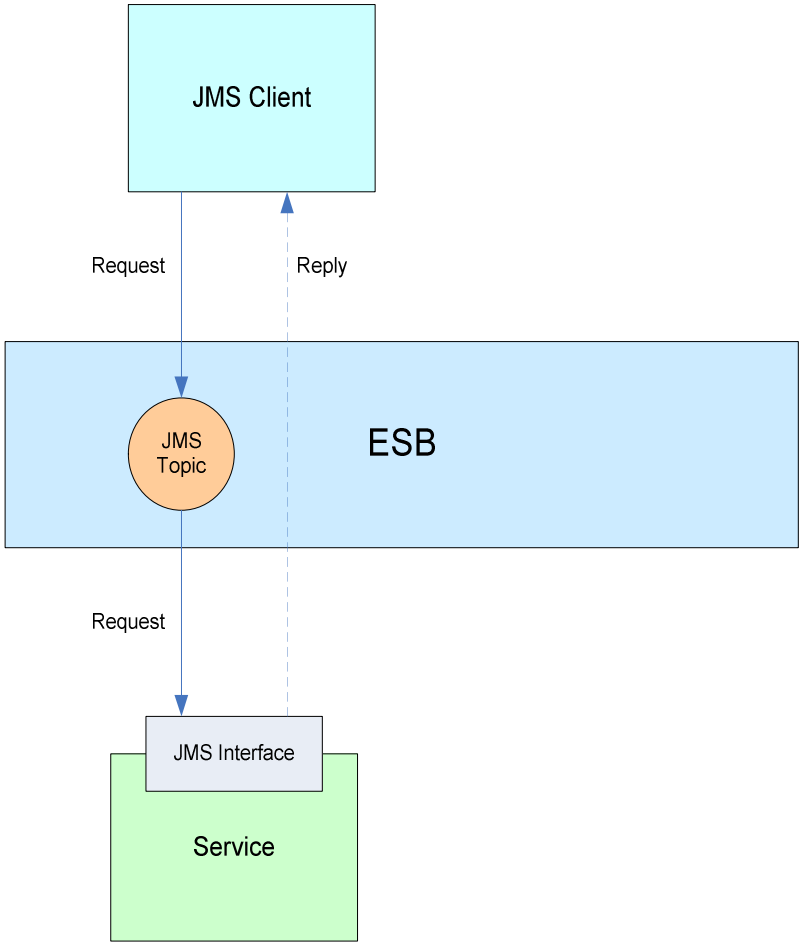
Basic Web Service Pattern

- Client issues request to a Web Service Interface
- Interface is defined using a WSDL
- Client expects one of several results:
  - Request is successfully processed and reply is returned in a timely manner
  - Request is accepted but reply returns an application level error
  - Request results in the return of a fault to the client
  - No reply or fault is returned in a timely manner



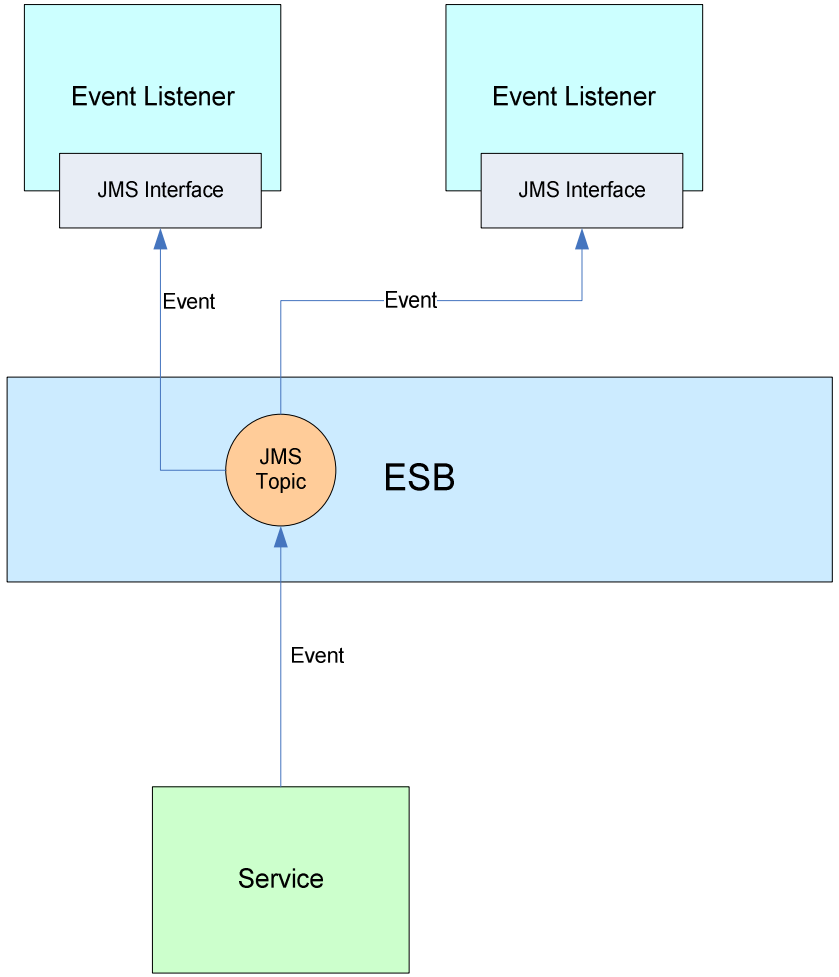
Basic JMS Request/Reply Pattern

- Client sends a JMS Message to a Topic or Queue
- Reply may be synchronous or asynchronous
- The reply is optional
- Client expects one of several results:
  - Request is successfully sent to a topic or queue and reply is returned in a timely manner
  - Request is accepted but reply returns an application level error
  - Attempt to send a Request to the topic or queue fails
  - No reply is ever received



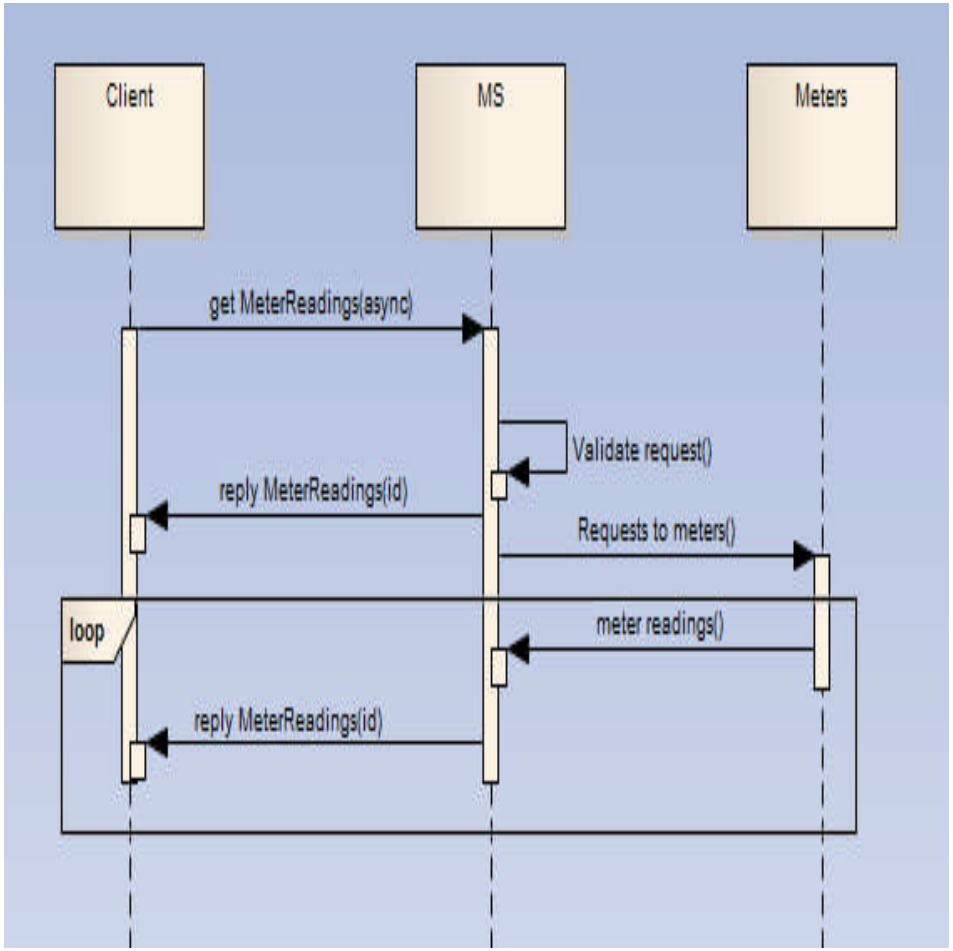
Event Listener Pattern

- A process listens for published events
- All subscribers to a topic will receive a copy of the message asynchronously
- JMS allows for durable subscriptions
- Event messages are send and consumed asynchronously



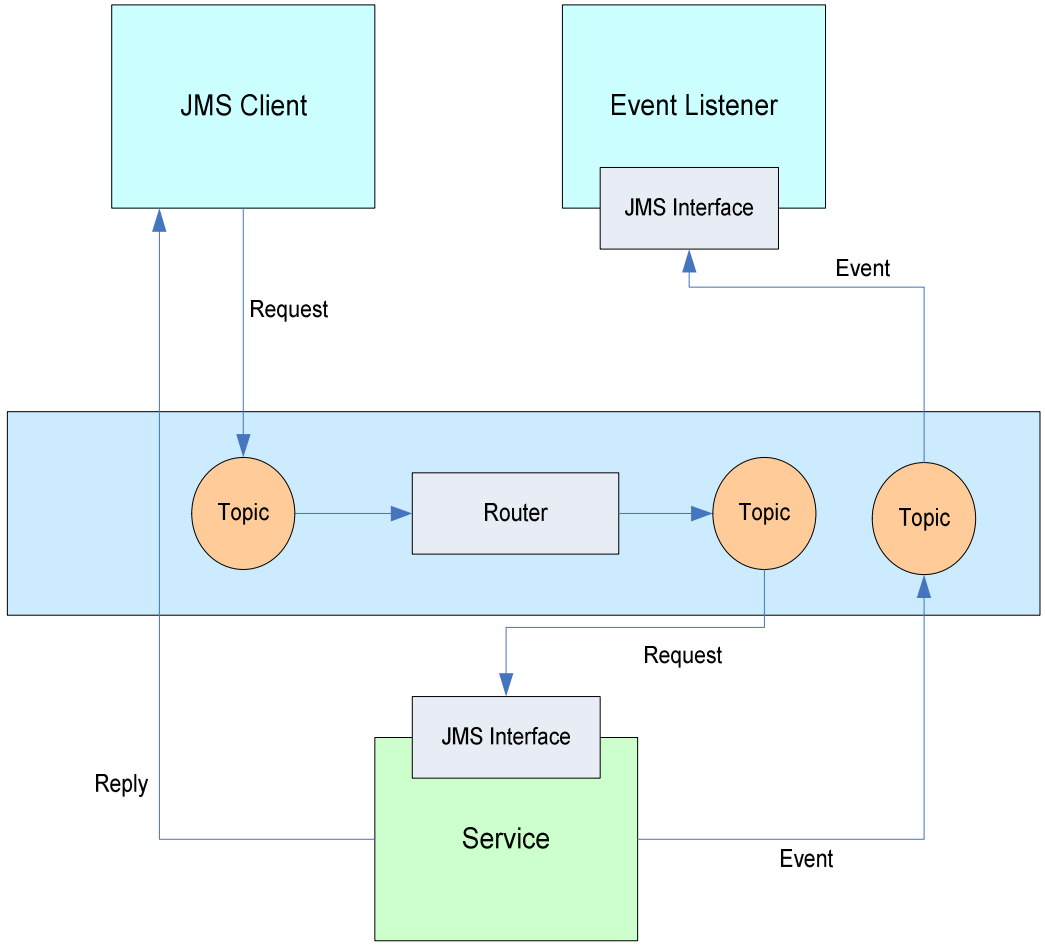
Asynchronous Request/Reply Pattern

- allows multiple replies to be returned to a requesting client asynchronously
- client makes a request to a service
- One use of this pattern is to obtain meter readings, where a metering system head end must request the desired information from one or more meters



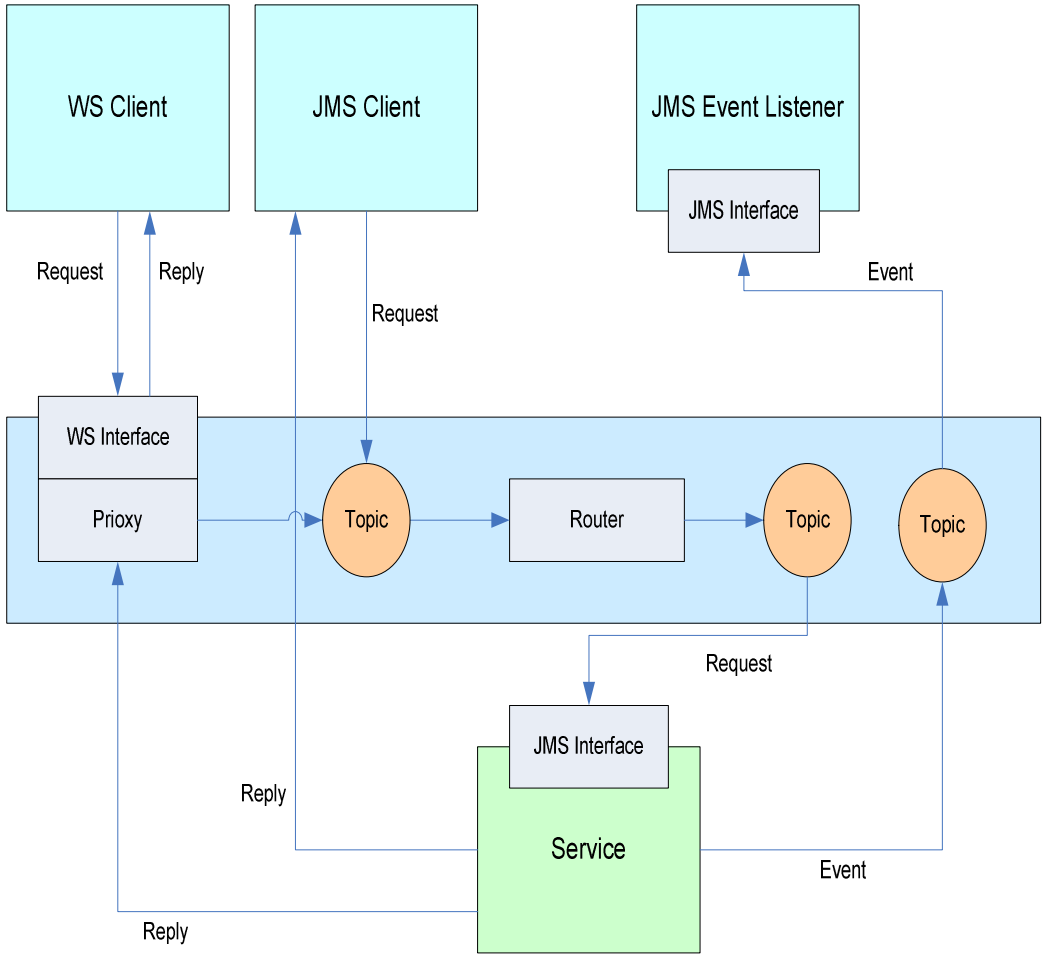
ESB Messaging Pattern Using JMS – ESB Content-Based Router

- ESB Pattern using JMA allow Routing of Request
- Bus can make decisions related to the request handling



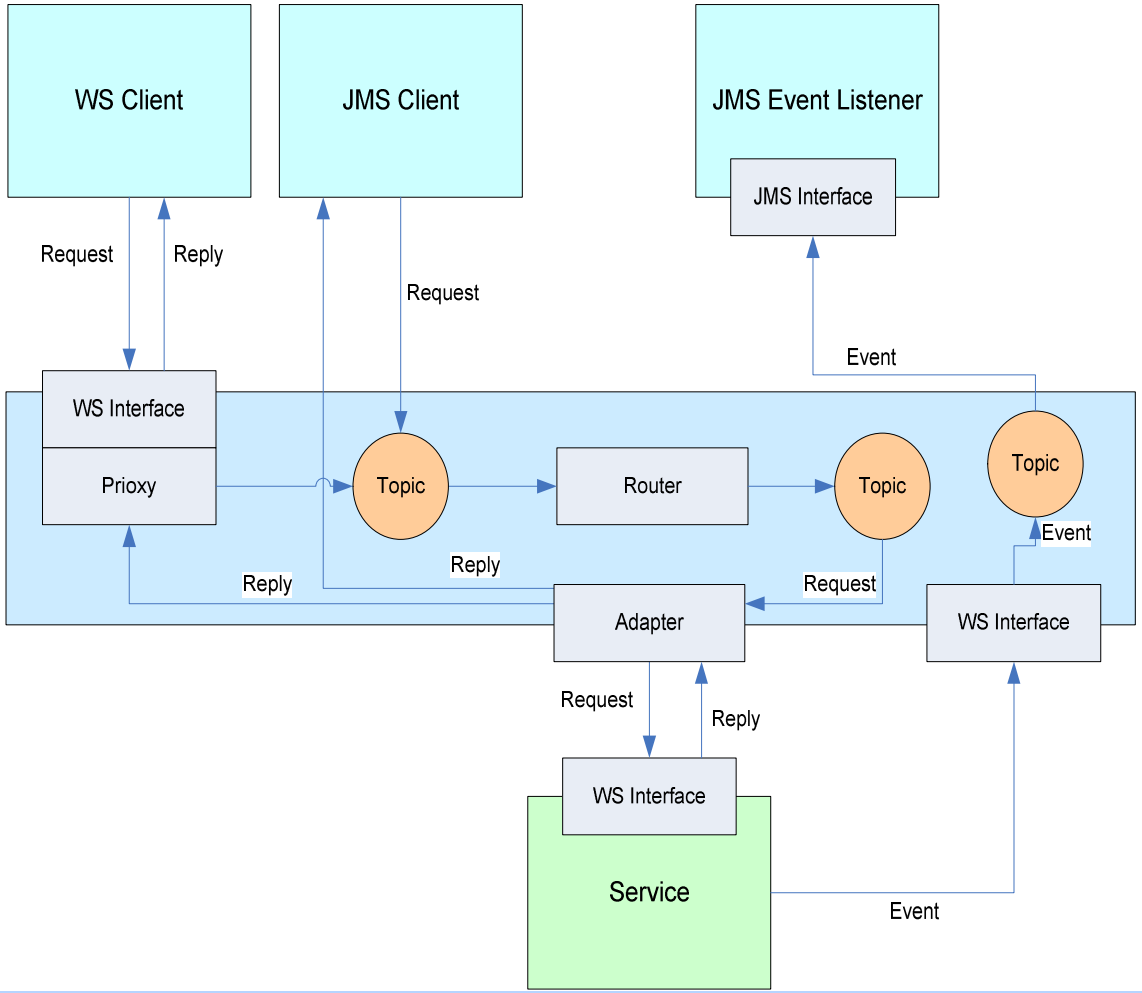
ESB Messaging Pattern Using Web Service Request – ESB with Smart Proxy and Content-Based Routing

- Extends the Content-based router to permit a request to be initiated by a Web Service or a JMS Client
- Smart Proxies can dispatch the requests and correlate the responses



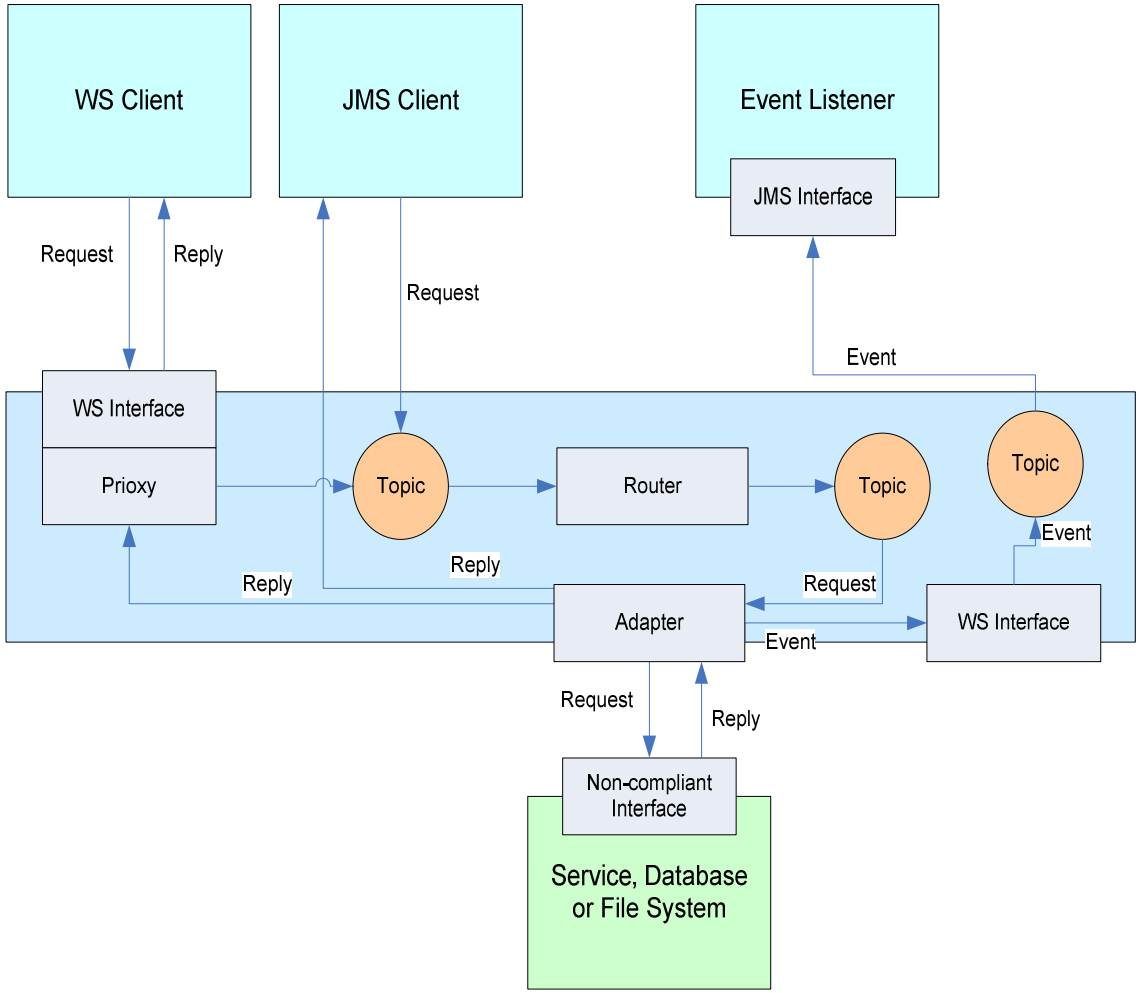
ESB Request Handling to Web Service – ESB with Proxies, Routers and Adapters

- Allows a service to expose its interface as a Web Service
- Adapter is in the ESB to convert the internal JMS message to the appropriate WS Request



ESB Request Handling via Adapter – ESB Integration to Non-Compliant Resources

- Have a Service that is not compliant to the standard
- Compliant Middleware Adapter is used in the ESB to provide a compliant interface



Custom Integration Patterns

- These would potentially include, but not be limited to patterns such as:
  - ***Content-Based Router***, where messages are routed based upon message content typically referenced using XPath expressions
  - ***Smart Proxy***, where messages may be redispatched to a specific destination service, where replies are accepted from the service and passed back to the client
  - ***Claim Check***, where a copy of an often very large file is maintained as a document for use by other processes, where the current status of the document is tracked, but the document is typically transported by means other than messaging
  - ***Transformation***, where transformations usually defined by XSL are used to reformat message contents
  - ***Bridge***, where a message published on a topic or queue may be forwarded to or received from another messaging infrastructure (this pattern can sometimes be implemented using third party products or simply through configuration)

Other Topics

- Message Organization – Provides an overview of the message organization as defined in the Part 9 standard.
- Interface Specifications – Gives a description of the WSDL Structure, document style SOAP binding, and MTOM (Message Transmission Optimization Mechanism)
- ESB Considerations – Describes common ESB characteristics and recommendations
- Security Considerations – Discusses XML Encrytion and XML Signatures.

Questions & Contacts

- Margaret Goodrich –
  - Home Office: 903-489-1494
  - Cell: 903-477-7176
  - Email: margaret@sisconet.com