Enterprise Integration Using IEC 61968-100

Margaret Goodrich, SISCO
Scott Neumann, UISOL
Introduction

- The purpose of this presentation is to describe enterprise integration using the IEC 61968-100 standard
- IEC 61968-100 is reflective of best practices and integration experiences, and has effectively been in use since 2006
- Approach published several years ago as an EPRI report
- IEC 61968-100 was formally approved in 2013
Integration Choices

- Depending upon the integration scenario, there are many factors that may drive the implementation of an information flow
- Some of the implementation options include:
  - Transactions through an ESB to create, change or delete objects
  - Event notification messages through an ESB that indicate the create, change or delete of objects
  - Messages through an ESB that support query requests, potentially with asynchronous replies
  - Web service requests for queries or transactions
  - Data set exchanges via CIM/XML files
  - Data set updates using incremental CIM/XML
  - Information transfer to a data warehouse using ETL mechanisms
  - HTTP requests
- There are also a wide variety of technical, performance, availability and security considerations that would need to be factored/addressed for any integration choice
IEC 61968-100 focuses on message-based integration using transport technologies such as JMS and web services.
Integration Patterns

• There are many possible integration patterns, but there are several that provide basic building blocks

• These basic patterns include:
  – Synchronous request/reply
  – Point to point
  – Publish/subscribe
  – Asynchronous request/reply

• Request/reply patterns can be used to support queries as well as transactions
Basic Integration Patterns

Synchronous Request/Reply

Callback

This is the sort of stuff you can do with even the most basic RPC and functionally limited technologies. Modern integration needs more than this, which is why we see ESBs.
Basic ESB Integration Patterns

Synchronous Request/Reply

Point-to-Point (One Way)

Publish/Subscribe

Asynchronous Request/Reply
Request/Reply Implementations

- Basic examples using web services and JMS
- Web services can either have generic interfaces or be strongly typed
- JMS can use topics or queues
Events

• Events are used to notify potentially interested processes that something has happened
• Events are best communicated using a publish/subscribe infrastructure such as a JMS implementation
• Web services need an intermediary service to distribute events
Integration Using an ESB

- Target service uses JMS
- Could be topics or queues
Another Integration Example

- Target service uses web services
- Targets could use other integration technologies as well
Common Message Envelope

- IEC 61968-100 defines a general approach for transport independent messages that minimally consist of a verb, noun and payload
- IEC 61968-100 defines a common message envelope (CME) in the form of an XML schema
- Messages using the CME can be wrapped within other wrappers (e.g. SOAP, WSDL, JMS)
- Usage of different parts of the CME (e.g. Request, Reply Payload) are dependent upon messaging pattern and verb
- Payload is defined using an XML Schema ‘any’
- Standard XSD provided as Message.xsd
Message Example

• This example wraps a CME inside the SOAP body
• This example shows what would be seen ‘on the wire’
• You can’t tell if this message was generated using a WSDL or not!
• The CME even allows for payloads that do not have a corresponding XSD
• This example uses SOAP, but the CME is in no way dependent upon SOAP

```xml
<soapenv:Envelope>
  <soapenv:Header>
  </soapenv:Header>
  <soapenv:Body>
    <ns:Message>
      <ns:Header>
        <ns:Verb>created</ns:Verb>
        <ns:Noun>MyBusinessObject</ns:Noun>
        <ns:Source>SourceApp</ns:Source>
        <ns:MessageID>35337</ns:MessageID>
      </ns:Header>
      <ns:Payload>
        <bo:MyBusinessObject>
          <bo:stuff>1</bo:stuff>
          <bo:moreStuff>More stuff</bo:moreStuff>
        </bo:MyBusinessObject>
      </ns:Payload>
    </ns:Message>
  </soapenv:Body>
</soapenv:Envelope>
```
Verbs

- Verbs support queries, transactions and events
- Get verb is used for queries
- Create, change, cancel, close and delete verbs are used for transactions
- Reply verb is used to respond to query and transaction requests
- Past tense verbs are used to indicate events
- Execute verb is used for a special case of complex transactions
- See IEC 61968-100 Annex B

![Diagram of Enterprise Service Bus using IEC 61968-100 with verbs and state transitions](image-url)
## Verb Usage

<table>
<thead>
<tr>
<th>Request Verb</th>
<th>Reply Verb</th>
<th>Event Verb</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>reply</td>
<td>(none)</td>
<td>query</td>
</tr>
<tr>
<td>create</td>
<td>reply</td>
<td>created</td>
<td>transaction</td>
</tr>
<tr>
<td>change</td>
<td>reply</td>
<td>changed</td>
<td>transaction</td>
</tr>
<tr>
<td>cancel</td>
<td>reply</td>
<td>canceled</td>
<td>transaction</td>
</tr>
<tr>
<td>close</td>
<td>reply</td>
<td>closed</td>
<td>transaction</td>
</tr>
<tr>
<td>delete</td>
<td>reply</td>
<td>deleted</td>
<td>transaction</td>
</tr>
<tr>
<td>execute</td>
<td>reply</td>
<td>executed</td>
<td>transaction</td>
</tr>
</tbody>
</table>
Patterns and Verbs

• **Queries** are indicated by the use of a ‘get’ verb on the request in a request/reply pattern, where the payload of the response message should return the requested information

• **Transactions** are indicated by the use of verbs such as ‘create’, ‘change’, ‘close’, ‘cancel’, ‘delete’, and could use request/reply or point-to-point patterns

• Both queries and transactions may result in responses (verb=‘reply’), which may indicate success or errors

• **Events** are indications that something has happened of potential interest, using past tense verbs such as ‘created’, ‘changed’, ‘closed’, ‘canceled’, ‘deleted’ and typically use a multicast pattern
Message Stereotypes

- There are several message stereotypes, where each convey a subset of the full Message envelope.
- Each of these are applied to different integration patterns.
- Request messages provide the ability to specify request parameters for requests that represent queries (i.e. verb=get).
- Request messages that represent transactions will typically supply payloads.
- Response messages to query requests will also supply payloads unless an error is encountered.
- Except for Fault messages, all require a header (since in a fault condition it may not have been possible in interpreting the header of an initiating request).
- These stereotypes are often used for defining web service interfaces where WS-I compliance is an issue, but are not necessary for less restrictive (more flexible) messaging transports such as JMS.
Nouns

• Nouns identify the type of the payload
• Many nouns are defined by parts of IEC 61968 (e.g. IEC 61968-9 for integration of metering systems)
• Nouns may also be defined as needed for products and custom integration projects
• The data related to a noun is typically, but not always, defined using an XML schema
• Other XML forms and Non-XML forms are also supported
Noun Example
Example with Verbs and Nouns

LMS

LDC

MDM

get(MeterReadings)

reply(MeterReadings)

create(EndDeviceControls)

reply(EndDeviceControls)

Perform load control

created(EndDeviceEvents)

created(EndDeviceEvents)
Message Correlation

- CorrelationID is set on initial request
- Initial CorrelationID is returned on related messages
Transactions
Transports

- IEC 61968 is by definition transport independent, but the reality is that a wide range of messaging technologies have largely given way to JMS and HTTP.
- JMS provides a powerful messaging API, where the worst thing about JMS is the name, as it is not limited to use by Java applications.
- HTTP and JMS have some trade-offs:
  - JMS provides a reliable transport with the capability to queue and multicast messages.
  - HTTP provides a standard ‘on the wire’ protocol, but JMS does not.
- SOAP may be layered on both of these transports, where the IEC 61968 message envelope may be placed within the SOAP body.
- When SOAP is used, it can be possible to define a WSDL that also provides a description of the interface.
- Other transports can also be considered, such as AMQP, XMPP and REST.
- ESBs support the use of intermediaries.
- IEC 61968-100 is being drafted to define specific, transport-dependent implementation profiles and integration patterns.
Web Services

- IEC 61968 messages can be ‘wrapped’ by a WSDL
- Two basic approaches: Loose coupled (generic, coarse grained) or tightly coupled (strongly typed, fine grained)
- For broadest compatibility, WSDL should be WS-I compliant and use document-wrapped style (still can do either loose or tightly coupled)
- Remember that HTTP is not the only transport possible... as JMS can be used to provide delivery guarantees, multicasting and other features not possible with HTTP
- The use of a SOAP envelope enables the use of many security tools
Generic vs. Strongly-Typed Web Services

- IEC 61968-100 describes two approaches to web services: generic (described in Annex D) and strongly-typed (described in Annex C).
- Generic web services are supported using a single, common WSDL called Generic.wsdl that has three operations (Request, Response, PublishEvent) that can be used for any noun without modification.
- Strongly-typed web services require the definition of noun-specific variants of Message.xsd and WSDLs, where sets of operations are defined for each noun.
- There are trade-offs to both approaches, in terms of flexibility vs. specificity, software engineering, complexity, etc.
Names

• Object naming is an important aspect of integration that is often overlooked especially where all systems do not use common mRIDs

• Issues:
  – The same object may be known in different system by different names
  – Messages need to use an identified that is known by source and targets
  – Some systems may need to be aware of multiple identifiers for a single object (e.g. local ID and external ID)

• Sources and targets MUST agree upon names to be used, or rely upon the use of bilateral mapping tables for object names

• A naming service could be used for allocating and managing names

• The performance implications of name lookups needs to be carefully considered

• Name class introduced in CIM v15 to help address these issues, providing a more realistic and useful model of object naming

• In theory, an mRID is just another type of name!
CIM Name Class

- Identified Objects can have many Names
- Each Name.name instance can be associated with a different NameType
- NameTypes can be associated with a NameTypeAuthority that can be responsible for allocating Name.name values within a NameType
- IdentifiedObject.name preserved for legacy reasons
Names in Messages

- Common usage of Name class in a message XSD
- NameType often option, but should be provided by source if more than one Name instance provided
- Target process will map the object using one of the names, as the name is either used locally or defined within a bilateral mapping table
Name Mapping Example

- Source sends a message related to object ‘ABC’
- Within message Name.name value is set to ‘ABC’
- Name.NameType.name can be used to specifically identify the type of name

<table>
<thead>
<tr>
<th>External ID</th>
<th>Local ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>1</td>
</tr>
<tr>
<td>XYZ</td>
<td>2</td>
</tr>
<tr>
<td>RTY</td>
<td>345</td>
</tr>
<tr>
<td>INB</td>
<td>67433</td>
</tr>
<tr>
<td>MJG</td>
<td>222</td>
</tr>
<tr>
<td>ECV</td>
<td>555</td>
</tr>
<tr>
<td>GGG</td>
<td>225</td>
</tr>
</tbody>
</table>

- Target receives the message
- Given that the Name.name value is unknown, a table lookup is required to know that object named ‘ABC’ as provided by the external system is equivalent to locally named object ‘1’
- This is a bilateral agreement required for integration where all systems do not share a globally unique ID

Note: This is one example, there are many options
More Information

• IEC TC57 Sharepoint (for CIM UG and IEC TC57 members): http://iectc57.ucaiug.org
• IEC Web Site: http://www.iec.ch
• E-mail: margaret@sisconet.com
• E-mail: sneumann@uisol.com

Slides Courtesy of Scott Neumann of UISOL