processing terminates. This step can also consider checks for authentication and authorization.

6) The service performs the desired processing, parsing the payload as needed. The service may parse the payload using an appropriate XML schema (as would define the payload type described by the message noun), using XPath expressions, using XSL transformations or other mechanisms. The service may also consider parameters provided in the message header and request packages for processing.

7) A response message is constructed, where a payload is rendered of the type identified by the noun as needed. This would commonly be the case in response to a 'get' request. The message reply element should be used to convey either a Result of 'OK' or an error code as appropriate.

8) The response message is returned to the client. This can use web services, JMS or other transport technologies as expected by the client.

9) The client processes the response message, parsing the payload as needed. The client should examine the reply element of the message to see if the request was successful (i.e. Result='OK') or encountered one or more errors.

10) Processing is completed

It is important to note that there are a variety of failure scenarios that can occur between steps 2 to 8, where a client should be able to handle a fault or time out when waiting for the reply to a request.

6.8 Event processing

An event message is a message that is published by a service (or more generally any event publisher) to potentially many listeners. Events may also be referred to as 'notifications'. Event listeners are consumers that have subscribed to one or more JMS topics of potential interest. In the case of web services, there would be an intermediary to send the events to subscribers listening via web services.

The basic sequence of event processing is as follows:

1) A service constructs an event message using a common message envelope and specifying a verb and a noun. The noun identifies the type of the payload, although not all event messages will have a payload.

2) The service sends the event message to the appropriate JMS topic. The ESB implementation can also transparently use intermediaries such as routers and adapters to transmit the event to the appropriate event listeners.

3) The listener accepts the message.

4) If the event message is invalid (e.g. incomplete, XML not well formed, etc.), processing terminates (typically after an error is logged).

5) The listener looks at the verb and noun combination, determining if the event can (or should) be processed, if not, processing terminates.

6) The listener performs the desired processing, parsing the payload as needed. The service may parse the payload using an appropriate XML schema (as would define the payload type described by the message noun), using XPath expressions, using XSL transformations or other mechanisms.

When transactional request messages (see 4.5) are processed on the bus that use the verb 'create', 'change, 'close', 'cancel' or 'delete', a corresponding event should be published using the corresponding past tense verb (e.g. 'created', 'changed, 'closed', 'canceled', 'deleted' or 'executed'). This should be issued after successful execution of the transaction by either the service that processed the request or a bus component. In these cases, the payload is identical to the payload used for the corresponding request message used to invoke the transaction.

The delivery guarantees are a consequence of the definition of the specific JMS topic or queue, as well as the means by which a listener subscribes (e.g. durable subscription).

6.9 Message correlation

One important aspect of asynchronous messaging patterns is the need to be able to correlate a request with a reply. The Header.CorrelationID is key to the association of requests with asynchronous replies. The following rules should be applied to messages to allow necessary 'linking':

- When a client provides a CorrelationID on a request, the value should be a either a hexadecimal UUID (e.g. 'D921A053-80C1-4DB6-960E-2603127B7B92') or a generated sequence number (e.g. 100023, 100024, 100025,...) that is effectively unique within the client making the request.
- If a request message includes a CorrelationID, the response message should return the same CorrelationID.
- If no CorrelationID is provided on a request message but a MessageID is provided, the response message should set the CorrelationID to the value of the MessageID that was provided on the request. MessageID should also be UUID or generated sequence numbers.
- If no MessageID or CorrelationID is provided on a request message, there is no way to correlate an asynchronous response to a specific request. Consequentially the CorrelationID cannot be set in the response message in a manner that identifies a linkage to a specific request.
- If a service is generating events as a direct consequence of a specific request, the CorrelationID should be set on the corresponding event message as per the previous rules if possible, noting that this may not always be possible and it is therefore not a requirement. This would provide a correlation between the event and the transaction that caused it.

Refer to the discussion and CorrelationID usage example provided in 5.2.5.

6.10 Complex transaction processing using OperationSet

6.10.1 General

The purpose of Subclause 6.10 is to describe the use of the OperationSet element provided by Message.xsd. This provides support for transactions. The Message.xsd message envelope has been extended to accommodate an OperationSet construct in both the payload and reply portions of Message.xsd. The OperationSet element is shown within Figure 39.

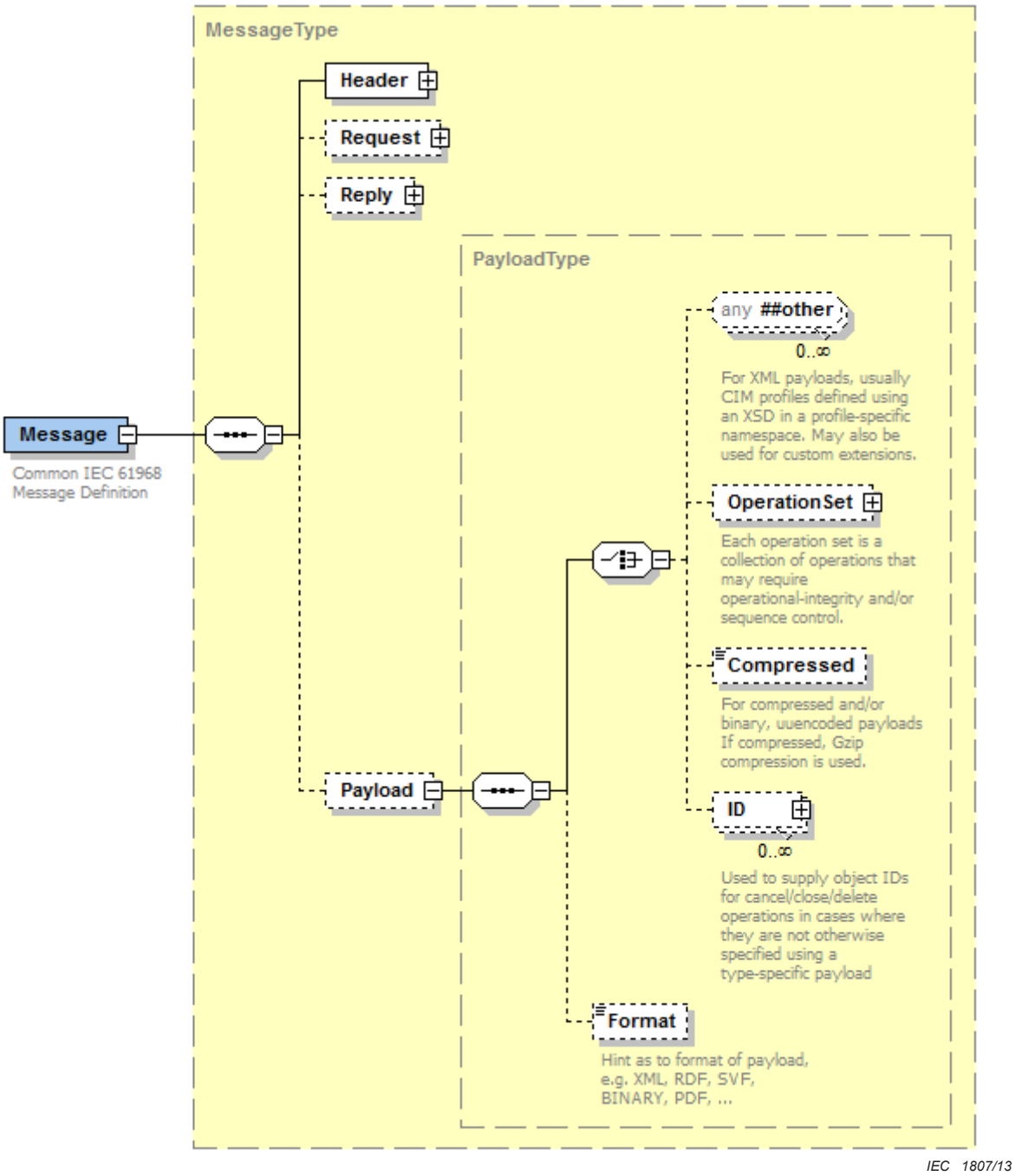

Figure 39 – Message OperationSet Element

There are two circumstances where the use of OperationSet might be necessary:

a) When modifying the configuration of a CIM object and the modification involves deleting one or more attributes or one or more instances of associated CIM objects. An example is removing a Register configuration from a Meter.

b) When performing two or more related actions that must be handled in a specific sequence and/or with overall transactional integrity (i.e., either all actions must succeed or all must be rolled back).

A message utilizing the OperationSet construct always has a Header verb of either ‘execute’ or ‘executed’ and a noun of ‘OperationSet’. An OperationSet in turn contains one or more Operation elements, and each OperationSet.Operation has an operationId which supplements the overall message CorrelationID to provide a fine-grained ability to correlate the contents of one or more reply messages with the individual operations in an OperationSet. Individual

Operation elements within an OperationSet have OperationSet-level verbs and nouns. Allowable verbs are create, created, change, changed, delete and deleted.

To support circumstance a) above, each Operation in an OperationSet also includes an elementOperation boolean. This Boolean is to be set to ‘true’ when the Operation verb is either ‘delete’ or ‘deleted’ and the intent is to delete individual attributes or individual instances of associated CIM classes from the object specified by the OperationSet noun (as opposed to deleting the entire CIM object specified by the Operation noun. If omitted, elementOperation is assumed to be ‘false’. It is emphasized that in this case, use of the Operation verb “delete” or “deleted” in combination with an elementOperation boolean set to ‘true’ effectively modifies (and does not delete) the CIM object specified by the Operation noun.

To support circumstance b) above, each OperationSet may have either an enforceMsgSequence boolean or an enforceTransactionalIntegrity boolean, or both. The enforceMsgSequence Boolean is to be set to ‘true’ when the Operations in the Operation set shall be executed in ascending order of their operationID. The enforceTransactionalIntegrity boolean is to be set to ‘true’ if all Operations in the OperationSet shall succeed. In this case, if all such Operations do not succeed, all shall be rolled back. If either or both of these booleans are omitted, they are assumed to be ‘false’.

When modifying the configuration of a CIM object using any of the verbs ‘change’, ‘changed’, ‘delete’ or ‘deleted’, only the ID of the object being changed and the information that is being changed is to be included. This is true whether or not an OperationSet is being used. It is for this reason that almost all elements within the IEC 61968-9 Master Data Management Profiles are optional in the profiles.

It is recommended that only one OperationSet be used, as multiple OperationSets would place more burden upon consumers and potentially involve unnecessarily large messages.

It should also be noted that while this provides the means to convey transactions using XML schema-based data structures, it is also technically possible to leverage IEC 61970-552[1] for transactions based upon RDF.

6.10.2 OperationSet Element

Figure 40 describes the OperationSet element in more detail. An OperationSet can:

- Require that each operation is sequentially executed by setting the enforceMsgSequence flag to ‘true’
- Require that transactional integrity be maintained (i.e. all or nothing), by setting the enforceTransactionalIntegrity flag to ‘true’
- Have one or more Operations, where each operation has a noun, verb and payload.

[1] To be published.

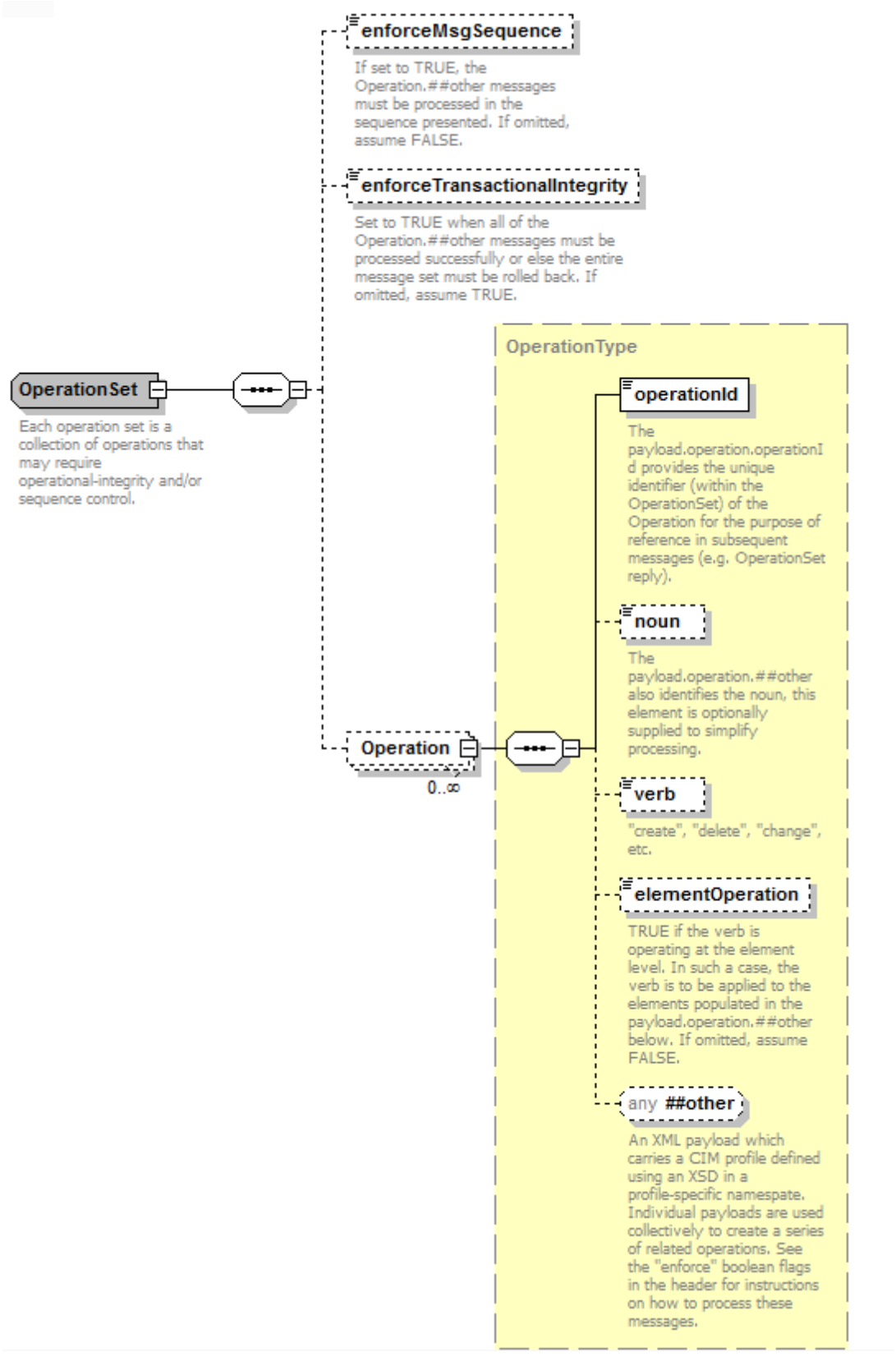

Figure 40 – OperationSet details

Within the Operation element, the noun will identify the type of the any element. The elementOperation value will cause the transaction to either act upon the object or elements within the object. Examples provided will further describe usage.

6.10.3 Patterns

Any given transaction may be executed using either a request-response message pattern (Request stereotype and Response stereotype messages) or a published event message pattern (Event stereotype messages). The four example sequence diagrams of Figure 41 illustrate the possible variations.

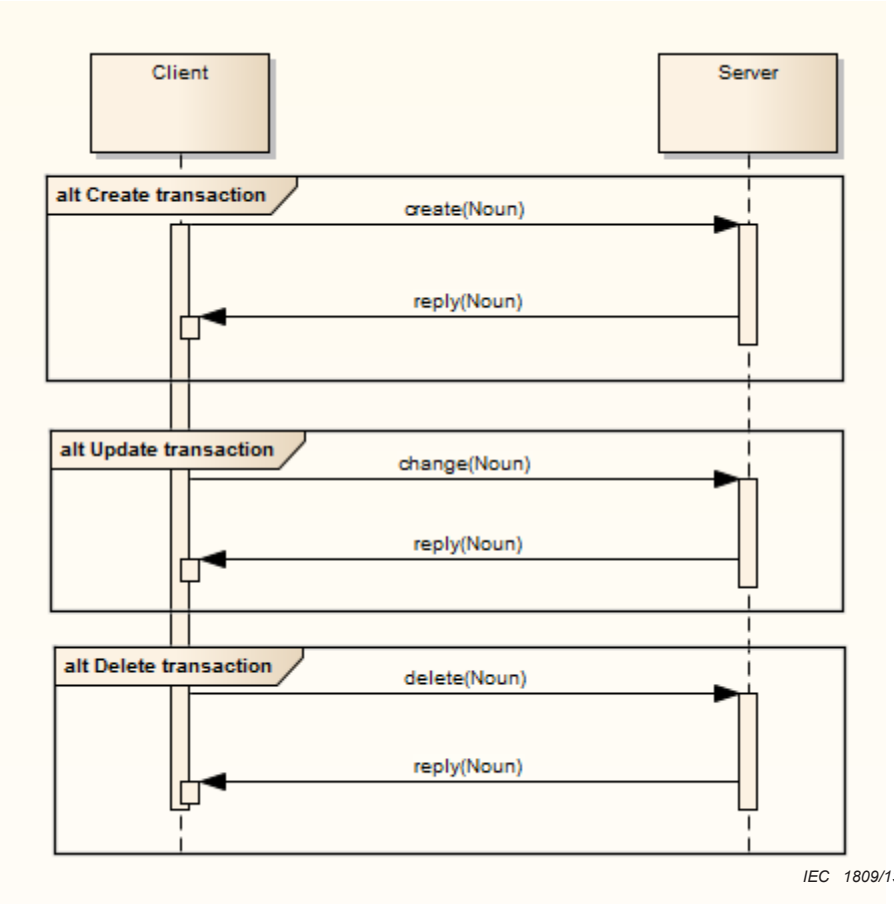

Figure 41 – Transactional Request/Response (non-OperationSet)

This request / response pattern can be used for transactions. Allowable verbs are ‘create’, ‘change’ and ‘delete’. Depending upon the scenario, there can be multiple replies to a given ‘create’, ‘change’, or ‘delete’ message. For example, a single create message can be issued to create multiple meters. In this case, the responding system can send a single reply message for all meters or multiple reply messages with the reply data for one or more meters in each message.

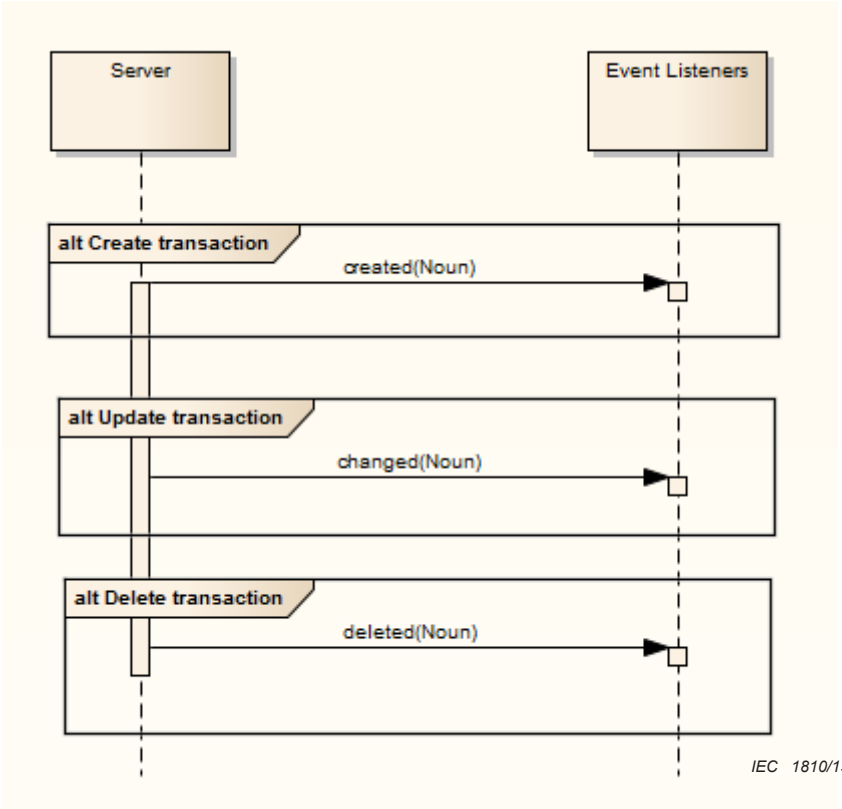

Figure 42 – Published events (non-OperationSet)

The published event pattern can also be used for transactions, as shown by the sequence diagrams of Figure 42. Allowable verbs are ‘created’, ‘changed’ and ‘deleted’. Using this pattern, an enterprise system may notify one or more other enterprise systems of events without requiring any acknowledgment or confirmation of successful processing.

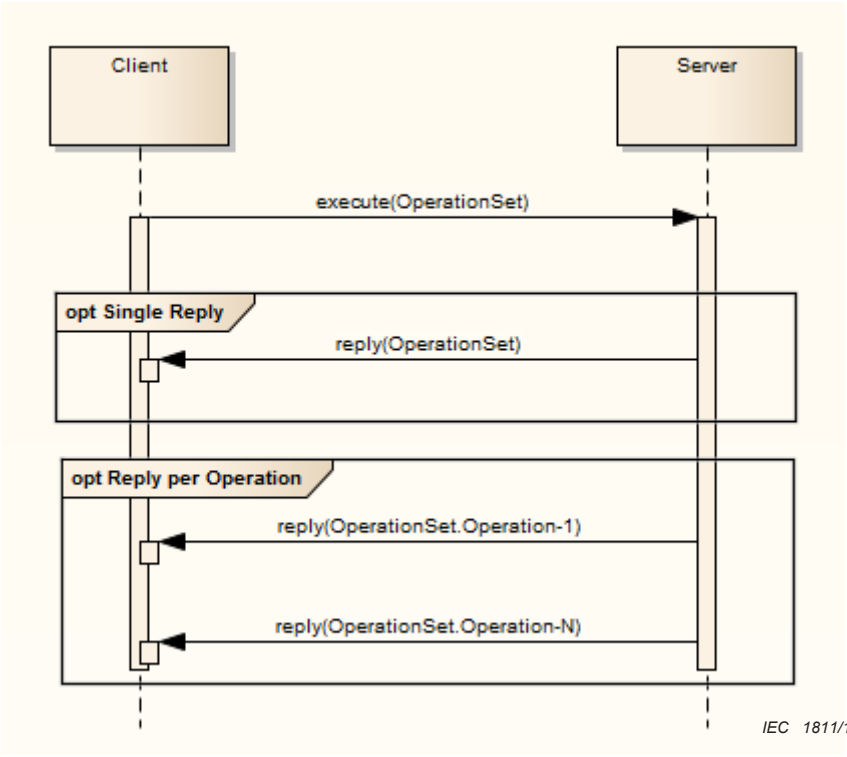

Figure 43 – Transactional Request/Response (OperationSet)

This request/response pattern can be used for any transaction involving an OperationSet, as shown by the sequence diagram of Figure 43. The verb in the message Header is always ‘execute’. The individual Operation(s) within the Operation set can have verbs and nouns consistent with the request / response transaction in Pattern 1. Depending upon the scenario, there can be multiple replies to a given execute / OperationSet transaction. For example, a single reply message can be sent for the entire OperationSet, or multiple reply messages can be sent, each with the reply data for one or more Operations in each message. The operationID element for each Operation in the request message is supplied in the reply message(s). This is used, in conjunction with the overall CorrelationID in the message Header(s) to correlate replies with their corresponding requests.

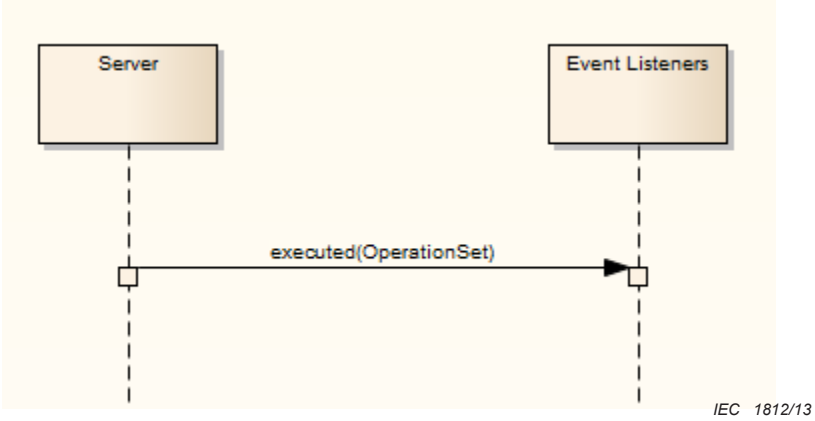

Figure 44 – Published event (OperationSet)

The published event pattern can also be used for any transaction involving an OperationSet, as shown in the sequence diagram of Figure 44. The verb in the message Header is always “executed”. The individual Operation(s) within the Operation set can have verbs and nouns consistent with the published event transaction in Pattern 2. Using this pattern, an enterprise system may notify one or more other enterprise systems of OperationSet events without requiring any acknowledgment or confirmation of successful processing.

6.10.4 OperationSet example

The following XML provides an example of a complex transaction that uses the Payload.OperationSet element.

```
<?xml version="1.0" encoding="UTF-8"?>
<RequestMessage
    xmlns = "http://iec.ch/TC57/2011/schema/message"
    xmlns:m = "http://iec.ch/TC57/2011/MeterConfig#"
    xmlns:up = "http://iec.ch/TC57/2011/UsagePointConfig#"
    xmlns:mdlc = "http://iec.ch/TC57/2011/MasterDataLinkageConfig#"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation = "http://iec.ch/TC57/2011/schema/message Message.xsd">
    <Header>
        <Verb>execute</Verb>
        <Noun>OperationSet</Noun>
        <Revision>2.0</Revision>
        <Timestamp>2012-12-20T09:30:47Z</Timestamp>
        <Source>CIS</Source>
        <AckRequired>true</AckRequired>
        <MessageID>D921A053-80C1-4DB6-960E-2603127B7B92</MessageID>
        <CorrelationID>D921A053-80C1-4DB6-960E-2603127B7B92</CorrelationID>
    </Header>
    <Payload>
        <OperationSet>
            <enforceMsgSequence>true</enforceMsgSequence>
            <enforceTransactionalIntegrity>true</enforceTransactionalIntegrity>
            <Operation>
                <operationId>1</operationId>
                <noun>MeterConfig</noun>
                <verb>create</verb>
                <mdlc:MeterConfig>
```

```
                    <mdlc:Meter>
                        <mdlc:formNumber>2S</mdlc:formNumber>
                        <mdlc:ConfigurationEvents>
                            <mdlc:createdDateTime>2012-12-
20T09:30:47Z</mdlc:createdDateTime>
                            <mdlc:effectiveDateTime>2012-12-
21T00:00:00Z</mdlc:effectiveDateTime>
                            <mdlc:Names>
                                <mdlc:name>C34531</mdlc:name>
                                <mdlc:NameType>
                                    <mdlc:name>MeterBadgeNumber</mdlc:name>
                                    <mdlc:NameTypeAuthority>
                                        <mdlc:name>UtilityXYZ</mdlc:name>
                                    </mdlc:NameTypeAuthority>
                                </mdlc:NameType>
                            </mdlc:Names>
                        </mdlc:ConfigurationEvents>
                    </mdlc:Meter>
                </mdlc:MeterConfig>
            </Operation>
            <Operation>
                <operationId>2</operationId>
                <noun>UsagePointConfig</noun>
                <verb>create</verb>
                <up:UsagePointConfig>
                    <up:UsagePoint>
                        <up:amiBillingReady>amiCapable</up:amiBillingReady>
                        <up:connectionState>connected</up:connectionState>
                        <up:isSdp>true</up:isSdp>
                        <up:isVirtual>false</up:isVirtual>
                        <up:phaseCode>B</up:phaseCode>
                        <up:readCycle>ReadCycleJ</up:readCycle>
                        <up:ConfigurationEvents>
                            <up:createdDateTime>2012-12-
20T09:30:47Z</up:createdDateTime>
                            <up:effectiveDateTime>2012-12-
21T00:00:00Z</up:effectiveDateTime>
                        </up:ConfigurationEvents>
                        <up:Names>
                            <up:name>UP43639</up:name>
                            <up:NameType>
                                <up:name>ServiceDeliveryPointID</up:name>
                                <up:NameTypeAuthority>
                                    <up:name>UtilityXYZ</up:name>
                                </up:NameTypeAuthority>
                            </up:NameType>
                        </up:Names>
                    </up:UsagePoint>
                </up:UsagePointConfig>
            </Operation>
            <Operation>
                <operationId>3</operationId>
                <noun>MasterDataLinkageConfig</noun>
                <verb>create</verb>
                <mdlc:MasterDataLinkageConfig>
                    <mdlc:ConfigurationEvent>
                        <mdlc:createdDateTime>2012-12-
17T09:30:47Z</mdlc:createdDateTime>
                        <mdlc:effectiveDateTime>2012-12-
21T00:00:00Z</mdlc:effectiveDateTime>
                    </mdlc:ConfigurationEvent>
                    <mdlc:Meter>
                        <mdlc:Names>
                            <mdlc:name>C34531</mdlc:name>
                            <mdlc:NameType>
                                <mdlc:name>MeterBadgeNumber</mdlc:name>
                                <mdlc:NameTypeAuthority>
                                    <mdlc:name>UtilityXYZ</mdlc:name>
                                </mdlc:NameTypeAuthority>
                            </mdlc:NameType>
                        </mdlc:Names>
                    </mdlc:Meter>
                    <mdlc:UsagePoint>
                        <mdlc:Names>
                            <mdlc:name>UP43639</mdlc:name>
                            <mdlc:NameType>
                                <mdlc:name>ServiceDeliveryPointID</mdlc:name>
                                <mdlc:NameTypeAuthority>
```

```
                                    <mdlc:name>UtilityXYZ</mdlc:name>
                                </mdlc:NameTypeAuthority>
                            </mdlc:NameType>
                        </mdlc:Names>
                    </mdlc:UsagePoint>
                </mdlc:MasterDataLinkageConfig>
            </Operation>
        </OperationSet>
    </Payload>
</RequestMessage>
```

The example complex transaction has three operations that do the following:

- Perform a ‘create MeterConfig’
- Perform a ‘create UsagePointConfig’
- Performs ‘create MasterDataLinkageConfig’

The XML identifies namespaces for MeterConfig, UsagePointConfig and MasterDataLinkageConfig as defined by IEC 61968-9.

6.11 Representation of time

The ISO 8601 standard is used to define the representations of time values that are conveyed through interfaces. This avoids issues related to time zones and daylight savings time changes.

Timestamps in messages published by a server process should use a prevailing time, using the following example format: *2007-03-27T14:00:00-05:00* (as time changes from CDT to CST, the *-05:00* would change to *-06:00*).

Timestamps in messages sent by a client process could use any ISO 8601 compliant timestamp.

It is extremely important to note that the use of ISO 8601 timestamps within message definitions for the external interfaces defined by this document in no way constrains other representations of time that may include:

- User interfaces, where local time or market hours may be used as desired
- Reports, where reports would be generated using an appropriate local time
- Internal integration, where an application may internally require some other time structure

6.12 Other conventions and best practices

The following are other conventions that shall be followed by this specification:

- Within XML definitions, tags should be namespace qualified. For example, an XML tag of ‘<tag>’ should be prefixed by a specific namespace reference, e.g. ‘<ns:tag>’. This will help to eliminate ambiguity. *(Note that many examples in this document are not namespace qualified for brevity and to aid legibility)*
- Quantities should be expressed using SI units where appropriate.

6.13 Technical interoperability

Open standards are a key part of the strategy to achieve technical interoperability. Standards of particular interest include:

- W3C standards
- OASIS WS-* standards

- IEC Common Information Model and related standards (e.g. IEC 61970-301 and IEC 61968-11)
- Java Message Service

It is very important that the implementation of Web Service interfaces not be dependent upon any specific proprietary, third party products. Another key requirement is that implementation of web service clients shall be possible using both Java and .Net development tools.

6.14 Service level agreements

Different categories of services will have different service level agreements (SLAs). The SLAs for some services are directly impacted by the variability in the amount of data that can be transferred.

The response time periods specified for each interface covered by an SLA typically will vary to some degree, based upon factors such as network and system loading. Consequentially, each SLA should be stated in a manner such that each SLA will be honoured X% of the time where X is often in the range of 90 to 100%.

One use of SLAs is to identify timeout periods for request handling.

6.15 Auditing, monitoring and management

The ESB will typically have capabilities for auditing, monitoring and management. There may also be common services that are used for the implementation of integration components within the ESB. Example functionality would often include:

- Logging
- Generation of unique identifiers
- Generation of signatures
- User authentication and authorization
- Identification of on-line service instances (where there may be multiple instances)

7 Payload specifications

Each noun used in a message identifies a payload type. Payload types are typically derived from the IEC CIM or other semantic models. Payload types used by the parts of IEC 61968 are always derived from the IEC CIM and have design artefacts (e.g. XSDs) that describe their structure. Cases where XSDs are not required include:

- Messages using RDF payloads as defined by IEC 61968-13 and IEC 61970-452.
- Messages using payloads as defined by IEC 61970-453.
- Response messages from services that dynamically generate XML (as in the case of SQL XML result sets).
- Non-XML compressed and encoded payloads.
- Encoded binary data (where XML formatting is not efficient as in the case of ‘high speed data’)

If an XSD is not available to describe the payload, it is the responsibility of the sender and receiver(s) to agree upon the specific formatting.

The CIM logical information model is described as a set of UML packages. The diagram in Figure 45 shows the use of the CIM from the perspectives of UML modelling and generation of design artefacts needed by integration tools. It illustrates the relationships between information models and contextual profiles that are used in conjunction with assembly rules in order to derive design artefacts.