

DIN EN 61968-100

ICS 29.240.01; 35.240.99

Einsprüche bis 2013-06-22

Entwurf

**Integration von Anwendungen in Anlagen der Elektrizitätsversorgung –
Systemschnittstellen für Netzführung –
Teil 100: Umsetzungsprofile
(IEC 57/1244/CDV:2012);
Englische Fassung FprEN 61968-100:2012**

Application integration at electric utilities –
System interfaces for distribution management –
Part 100: Implementation profiles
(IEC 57/1244/CDV:2012);
English version FprEN 61968-100:2012

Intégration d'applications pour les services électriques –
Interfaces système pour la gestion de distribution –
Partie 100: Profils de mise en oeuvre
(CEI 57/1244/CDV:2012);
Version anglaise FprEN 61968-100:2012

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2013-04-22 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfes besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal des DIN unter www.entwuerfe.din.de bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an dke@din.de möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/stellungnahme oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (Hausanschrift: Stresemannallee 15, 60596 Frankfurt am Main).

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 113 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments FprEN 61968-100:2012 „Integration von Anwendungen in Anlagen der Elektrizitätsversorgung – Systemschnittstellen für Netzführung – Teil 100: Umsetzungsprofile“ (Entwurf in der Umfrage) ist unverändert in diesen Norm-Entwurf übernommen worden.

Die Internationale Elektrotechnische Kommission (IEC) und das Europäische Komitee für Elektrotechnische Normung (CENELEC) haben vereinbart, dass ein auf IEC-Ebene erarbeiteter Entwurf für eine Internationale Norm zeitgleich (parallel) bei IEC und CENELEC zur Umfrage (CDV-Stadium) und Abstimmung als FDIS (en: Final Draft International Standard) bzw. Schluss-Entwurf für eine Europäische Norm gestellt wird, um eine Beschleunigung und Straffung der Normungsarbeit zu erreichen. Dem entsprechend ist das internationale Dokument IEC 57/1244/CDV:2012 „Application integration at electric utilities – System interfaces for distribution management – Part 100: Implementation profiles“ unverändert in den Entwurf Englische Fassung FprEN 61968-100:2012 übernommen worden.

Das internationale Dokument wurde vom TC 57 „Power systems management and associated information exchange“ der Internationalen Elektrotechnischen Kommission (IEC) erarbeitet.

Bei der Abstimmung zu dem Europäischen Schluss-Entwurf bei CENELEC und dem Internationalen Schluss-Entwurf bei IEC [Final Draft International Standard (FDIS)] sind jeweils nur „JA/NEIN“-Entscheidungen möglich, wobei „NEIN“-Entscheidungen fundiert begründet werden müssen. Dokumente, die bei CENELEC als Europäische Norm angenommen und ratifiziert werden, sind unverändert als Deutsche Normen zu übernehmen.

Für dieses Dokument ist das nationale Arbeitsgremium K 952 „Netzleittechnik“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/2004 festgelegt, dass DIN-Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der Informationsverarbeitung gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Diese Bedingungen sind für den vorliegenden Normentwurf erfüllt.

Da sich die Benutzer des vorliegenden Normentwurfs der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 61968-100 veröffentlicht. Zu deren Abschnitt 3, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als informativer Nationaler Anhang NA des vorliegenden Normentwurfs hinzugefügt. Für viele der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Dieser Normentwurf steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Es ist vorgesehen, auch bei der entsprechenden zukünftigen Deutschen Norm auf die deutsche Sprachfassung zu verzichten und diese in der englischsprachigen Fassung zu veröffentlichen.

Nationaler Anhang NA (informativ)

3 Begriffe und Abkürzungen

3.1 Begriffe

3.1.1 Allgemeines

Für die Anwendung dieses Dokuments gelten die Begriffe nach IEC 60050-300, IEC 61968-2, IEC 62051, IEC 62055-31 und die folgenden Begriffe.

Wenn Abweichungen zwischen den Begriffen in der vorliegenden Norm und den Begriffen in den anderen angegebenen IEC-Normen bestehen, haben die in IEC 61968-2 definierten Begriffe Vorrang vor den anderen aufgeführten Begriffen und die im vorliegenden Dokument definierten Begriffe haben Vorrang vor den in IEC 61968-2 definierten Begriffen. Dieser Abschnitt enthält einige Definitionen, die Wikipedia entnommen wurden.

3.1.2

Unternehmensdienstebus

(en: enterprise service bus, ESB)

Ein Unternehmensdienstebus (ESB) bezieht sich auf ein Konstrukt einer Softwarearchitektur, das als Integrationsschicht benutzt wird. Dieses Konstrukt wird gewöhnlich durch Technologien in einer Kategorie von Infrastrukturprodukten der Middleware implementiert, in der Regel auf der Grundlage anerkannter Normen, die grundlegende Dienste für komplexere Architekturen über eine ereignisgesteuerte und normenbasierte Messaging-Engine (den Bus) zur Verfügung stellt.

Ein ESB stellt im Allgemeinen eine Abstraktionsschicht an der Spitze der Implementierung eines Nachrichtenübermittlungssystems eines Unternehmens zur Verfügung, mit der Integrationsarchitekten den Wert einer Nachrichtenübermittlung ohne das Schreiben eines Programms auswerten können. Im Gegensatz zum klassischen Ansatz der Unternehmensanwendungsintegration (en: enterprise application integration, EAI) eines monolithischen Stacks in einer Nabe-Speiche-Architektur (en: hub and spoke architecture) setzt sich ein Unternehmensdienstebus aus Grundfunktionen zusammen, die in ihre Bestandteile aufgelöst wurden, mit einer verteilten Verwendung, wo sie benötigt werden, und die soweit erforderlich harmonisch zusammenarbeiten.

Ein ESB implementiert keine diensteorientierte Architektur (SOA), sie stellt aber die Merkmale zur Verfügung, mit denen eine solche implementiert werden kann.

3.1.3

Java-Messaging-System (JMS)

Die Anwendungsprogramm-Schnittstelle des Java-Nachrichtendienstes ist eine Java-nachrichtenorientierte Middleware-API für das Senden von Nachrichten zwischen zwei oder mehr Clients. Das JMS unterstützt Anforderung/Antwort- (request/reply), Veröffentlichen/Abonnieren- (publish/subscribe) und Punkt-zu-Punkt-Protokolle der Nachrichtenübermittlung. Es ist Teil der Java-Plattform, Enterprise Edition, und wird durch eine Spezifikation definiert, die unter dem Java-Community-Prozess als JSR 914 entwickelt wurde. Es muss beachtet werden, dass einige ESB-Produktanbieter Sprachbindungen für JMS an C, C++ und/oder C# anbieten, wodurch die Bezeichnung JMS falsch wird. Wenn sich das Drahtprotokoll zwischen verschiedenen JMS-Implementierungen unterscheidet, ist es oft einfach, die verschiedenen Implementierungen miteinander zu verbinden.

3.1.4

Diensteorientierte Architektur

(en: service-oriented architecture, SOA)

Eine diensteorientierte Architektur (SOA) ist ein Architekturmuster eines Computersystems für die Erzeugung und Nutzung von Geschäftsprozessen, die während ihrer Lebensdauer als Dienste dargestellt werden. Die SOA definiert außerdem die IT-Infrastruktur und stellt diese bereit, damit verschiedene Anwendungen Daten austauschen und an den Geschäftsprozessen teilnehmen können. Diese Funktionen sind locker mit den

Betriebssystemen und Programmiersprachen verknüpft, die den Anwendungen zugrunde liegen. Die SOA unterteilt Funktionen in unterschiedliche Einheiten (Dienste), die über ein Netzwerk verteilt und zur Erzeugung von Geschäftsanwendungen zusammengestellt und wiederverwendet werden können. Diese Dienste kommunizieren miteinander durch die Übermittlung von Daten von einem Dienst zu einem anderen oder durch die Koordination einer Aktivität zwischen zwei oder mehr Diensten. SOA-Konzepte bauen häufig auf älteren Konzepten einer verteilten Verarbeitung und einer modularen Programmierung auf und haben sich aus diesen entwickelt.

3.1.5

Ereignisgesteuerte Architektur

(en: event-driven architecture, EDA)

Eine ereignisgesteuerte Architektur (EAD) ist ein Software-Architekturmuster zur Förderung der Erzeugung, Erkennung und Verarbeitung von Ereignissen. Ein Ereignis ist jede Zustandsänderung von möglichem Interesse. Innerhalb einer EDA werden gewöhnlich unter Anwendung von Veröffentlichen/Abonnieren-Nachrichtenübermittlungsmustern Ereignisse zwischen locker verknüpften Softwarekomponenten und Diensten übertragen. Die EDA ist auf die SOA in dem Maß abgestimmt, dass SOA 2.0 auch als „ereignisgesteuerte“ SOA bezeichnet wird. Die EDA bildet die Grundlage für eine Vielzahl von Geschäftsinformatikmustern (en: business intelligence) einschließlich komplexer Ereignisverarbeitungsmuster.

3.1.6

Simple Object Access Protocol (SOAP)

Das Simple Object Access Protocol (SOAP) ist eine Norm, die die Formatierung von XML-Nachrichten definiert. SOAP dient als Basisschicht für den Webdienst-Protokollstapel. SOAP wird auch häufig innerhalb des JMS benutzt. Der gebräuchliche Transport für das SOAP sind HTTP-, HTTPS- und anwendereigene JMS-Transporte. SOAP wird mitunter auch als „dienstorientiertes Architekturprotokoll“ bezeichnet. SOAP ist eine Empfehlung von W3C.

Zwei übliche Versionen sind SOAP 1.1 und SOAP 1.2. Neue Integrationen sollten möglichst SOAP 1.2 verwenden.

3.1.7

Webdienste

(en: Web Services, WS)

Ein Webdienst wird von W3C definiert als „ein zur Unterstützung interoperabler Maschine-zu-Maschine-Interaktionen über ein Netzwerk entworfenes Softwaresystem“. Webdienste sind häufig einfach Web-APIs wie das Internet, auf die über ein Netzwerk zugegriffen werden kann und die auf einem abgesetzten System ausgeführt werden, das die geforderten Dienste bereitstellt.

Die Definition von Webdiensten von W3C schließt viele verschiedene Systeme ein, im üblichen Gebrauch bezieht sich der Begriff aber auf Clients und Server, die mit XML-Nachrichten nach der SOAP-Norm kommunizieren. Sowohl für die praktische Anwendung als auch für die Terminologie liegt die Annahme zugrunde, dass es auch eine maschinenlesbare Beschreibung der vom Server unterstützten Operationen gibt, die in der Webdienste-Beschreibungssprache (WSDL) geschrieben ist. Diese Bedingung ist keine Anforderung eines SOAP-Endpunktes, sondern eine Vorbedingung für die automatisierte Client-seitige Codeerzeugung in vielen Java- und .Net-Entwicklungswerkzeugen.

Obwohl Webdienste zwei grundlegende Formate aufweisen, d. h. dokumentbezogen und RPC-bezogen sind, richtet sich die vorliegende Spezifikation auf das dokumentbezogene Format. Damit werden die SOA und der Transport von Nutzinformationen nach IEC 61968 unterstützt. Für eine verbesserte Interoperabilität ist die in Dokumente eingebundene Form erforderlich.

Es muss beachtet werden, dass Webdienste eine Veröffentlichen/Abonnieren-Nachrichtenübermittlung ohne Weiteres nicht unterstützen, sofern nicht Mechanismen wie WS-Eventing angewendet werden. Diese Spezifikation beschreibt jedoch auch einen Ansatz für den ESB für die Beförderung von Nachrichten asynchron zu den konfigurierten Abonnenten.

In der vorliegenden Norm werden zwei Ansätze für die Nutzung von Webdiensten diskutiert, bei denen die WSDL-Operationen entweder generisch oder stark typisiert sind.

3.1.8

Webdienste-Beschreibungssprache

(en: Web Services Definition Language, WSDL)

Die Webdienste-Beschreibungssprache (WSDL) ist eine XML-basierte Sprache zur Beschreibung von Webdiensten. WSDL wird oft zusammen mit dem SOAP und dem XML-Schema zur Bereitstellung von Webdiensten über das Internet (oder ein Intranet) benutzt. WSDL ist eine Empfehlung von W3C.

Zwei übliche Versionen sind WSDL 1.1 und WSDL 2.0. WSDL 2.0 unterstützt die Interoperabilität zwischen Java- und .Net-Implementierungen besser.

Die im vorliegenden Dokument angegebenen WSDL-Vorlagen beruhen auf WSDL 1.1 und damit stellt WSDL 1.1 eine Vorbedingung dar.

3.1.9

XML-Schema (XSD)

Das von W3C im Mai 2001 als Empfehlung veröffentlichte XML-Schema ist eine von mehreren XML-Schemensprachen. Es war die erste gesonderte Schemensprache für XML, die den Empfehlungsstatus von W3C erreicht hat.

Wie alle XML-Schemensprachen kann mit XML-Schema ein Schema ausgedrückt werden, d. h. eine Menge von Regeln, denen ein XML-Dokument entsprechen muss, um nach diesem Schema als „gültig“ angesehen zu werden. Im Gegensatz zu den meisten anderen Schemensprachen wurde XML-Schema jedoch auch mit der Absicht entwickelt, dass die Bestimmung der Gültigkeit eines Dokumentes auch eine Sammlung von Informationen erzeugen würde, die an bestimmte Datentypen gebunden sind. Eine Instanz von XML-Schema ist eine XML-Schemendefinition (XSD) mit der typischen Dateinamenerweiterung „.xsd“. Die Sprache selbst wird mitunter regelwidrig als XSD bezeichnet.

Es ist zu beachten, dass Nutzinformationen nach IEC 61968 anhand von XML-Schemen definiert sind. Solche XML-Schemen stellen normative Spezifikationen für Anwendungsnutzinformationen zur Verfügung, die nach dieser Norm befördert werden.

3.1.10

Representational State Transfer (REST)

Der Representational State Transfer (REST) ist eine Alternative zur Anwendung von SOAP-basierten Webdiensten. REST selbst ist gegenwärtig keine Norm, sondern stellt ein Architekturmuster dar.

REST fördert das HTTP, wobei jeder URL eine Darstellung eines Objektes ist. Schnittstellen können in Form von XML-Nutzinformationen für Anforderungen und Antworten definiert werden. Für die Anwendung von REST ist keine WSDL erforderlich. Darüber hinaus müssen die für Anforderungen und Antworten benutzten XML-Dokumente nicht unter Anwendung einer XSD definiert werden, obwohl es für die XML üblich ist, dass sie einer XSD entspricht.

Innerhalb von REST kann ein Objekt mit einem HTTP-GET abgerufen (gelesen) werden. Genauso wird ein HTTP-POST zur Erzeugung eines Objektes, ein HTTP-PUT zur Änderung eines Objektes und ein HTTP-DELETE zum Löschen eines Objektes benutzt.

REST wird hier aus Gründen der Vollständigkeit lediglich diskutiert, da er in einem Integrationsprojekt auftreten kann. Gegenwärtig gibt es jedoch keine bestimmten Empfehlungen für die Aufnahme in die vorliegende Norm. Dabei kann es sich um einen zukünftigen Anhang handeln.

3.1.11

Warteschlange

(en: queue)

Eine Warteschlange ist ein von vielen Nachrichtenübermittlungsprodukten unterstütztes Konstrukt, zur Sicherstellung einer zuverlässigen Nachrichtenübermittlung mit Zustellungsgewährleistung. Ein JMS stellt eine Norm-API zur Verfügung, das warteschlangenbasierte Nachrichtenübermittlungsmodelle bereitstellt. Außerdem definiert AMQP ein offenes Protokoll für die warteschlangenbasierte Nachrichtenübermittlung.

3.1.12

Thema (de: topic)

Ein Thema ist ein von vielen Nachrichtenübermittlungsprodukten unterstütztes Konstrukt, das Veröffentlichen/Abonnieren-Nachrichtenübermittlungsmuster ermöglicht, wobei es viele mögliche Abnehmer einer Nachricht geben kann, welche von einem Herausgeber zu einem benannten Thema gesendet worden ist. Themen werden gewöhnlich als Ziel für Ereignisnachrichten benutzt. Themen werden direkt vom JMS unterstützt.

3.1.13**Nachrichtenbestimmungsort****(en: message destination)**

Ein Nachrichtenbestimmungsort ist die Zieladresse für eine Nachricht, unabhängig davon, ob es sich um eine Anforderungs-, eine Antwort- oder eine Ereignisnachricht handelt. Bei Anwendung eines JMS kann der Bestimmungsort ein Thema oder eine Warteschlange sein. Bei der Anwendung von HTTP-Mechanismen wie Webdiensten wird der Bestimmungsort als ein URL angegeben.

3.1.14**Anforderung****(en: request)**

Eine Anforderung ist eine Nachricht, die von einem Client (oder einer Quelle) zu einem Server (oder einem Ziel) gesendet wird, wobei eine Antwort erwartet wird. Die Anforderung kann entweder eine Abfrage (bei der Daten vom Zielgerät zurückgesendet werden) oder eine Transaktion (bei der Daten im Zielgerät modifiziert werden) sein. Eine Anforderung benutzt Verben wie „get“ (erhalten), „create“ (erzeugen), „change“ (ändern), „delete“ (löschen), „cancel“ (aufheben), „close“ (schließen) oder „execute“ (ausführen).

3.1.15**Antwort****(en: response)**

Eine Antwort ist eine Nachricht, die als eine Folge einer Anforderung in der Regel vom Zielgerät der Anforderung zur Quelle der Anforderung gesendet wird. Antwortnachrichten sind gleichbedeutend mit „reply“-Nachrichten. Eine Antwortnachricht benutzt das Verb „reply“ (rückmelden).

3.1.16**Abfrage****(en: query)**

Eine Abfrage ist eine Anforderung, bei dem das Ziel erwartungsgemäß Informationen an die Quelle der Anforderung zurücksendet. Die angeforderte Nachricht für eine Abfrage benutzt das Verb „get“ (erhalten). Sie wird gewöhnlich mit einem Anfrage/Antwort-Muster implementiert.

3.1.17**Transaktion****(en: transaction)**

Eine Transaktion ist eine Anforderung, bei der das Ziel die zu verarbeitende Information gewöhnlich modifiziert. Eine Transaktionsanforderung benutzt Verben wie „create“ (erzeugen), „change“ (ändern), „delete“ (löschen), „cancel“ (aufheben), „close“ (schließen) oder „execute“ (ausführen). Sie kann mit einer Vielzahl von Mustern implementiert werden. Wird ein anderes Muster als Anforderung/Antwort benutzt, sollte eine Zustellungsgarantie durch den Transport gegeben werden.

3.1.18**Ereignis****(en: event)**

Der Begriff „Ereignis“ wird stark beansprucht und kann in verschiedenen Zusammenhängen verschiedene Bedeutungen haben. Die allgemeinste Definition eines Ereignisses ist „etwas, das an einem bestimmten Ort und zu einer bestimmten Zeit geschieht“ oder „eine Zustandsänderung von möglichem Interesse“. Als eine Folge eines Ereignisses kann „etwas“ eine „Ereignisnachricht“ erzeugen, um den Umstand zu melden, dass eine bestimmte Art eines Ereignisses zu einem bestimmten Zeitpunkt eingetreten ist. Ereignisnachrichten sind daher dem Wesen nach asynchron und werden gewöhnlich unter Anwendung einer themenbasierten Nachrichtenübermittlung an interessierte Abonnenten ausgegeben.

Ereignisnachrichten sind asynchrone Nachrichten. Eine Anwendung darf im Allgemeinen ein Ereignis erzeugen, wenn eine bestimmte Bedingung festgestellt wird oder eine Transaktion verarbeitet worden ist. Es gibt mehrere Integrationsmuster, die sich auf die Unterstützung von Ereignissen beziehen.

Ereignisse werden gewöhnlich asynchron als Ereignisnachrichten von einem System oder einer Anwendung herausgegeben, um bestimmte Bedingungen zu melden. In bestimmten Fällen kann ein System oder eine Anwendung intern eine bestimmte Bedingung feststellen, aber in anderen Fällen kann die Bedingung von einer äußeren Quelle wie einem Gerät festgestellt werden. Ereignisnachrichten benutzen Präteritumverben wie „created“ (erzeugt), „changed“ (geändert), „deleted“ (gelöscht), „closed“ (geschlossen), „canceled“ (aufgehoben) oder „executed“ (ausgeführt).

CONTENTS

1	Scope	7
1.1	Scope of IEC 61968	7
1.2	Scope of this Document	7
2	Normative References	9
3	Terms, definitions and abbreviations	10
3.1	Terms and definitions	10
3.1.1	General	10
3.1.2	Enterprise Service Bus	10
3.1.3	Java Messaging System (JMS)	10
3.1.4	Service-Oriented Architecture (SOA)	10
3.1.5	Event-Driven Architecture (EDA)	11
3.1.6	Simple Object Access Protocol (SOAP)	11
3.1.7	Web Services (WS)	11
3.1.8	Web Services Definition Language (WSDL)	11
3.1.9	XML Schema (XSD)	12
3.1.10	Representational State Transfer (REST)	12
3.1.11	Queue	12
3.1.12	Topic	12
3.1.13	Message Destination	12
3.1.14	Request	13
3.1.15	Response	13
3.1.16	Query	13
3.1.17	Transaction	13
3.1.18	Event	13
3.2	Abbreviations	13
4	Use Cases	15
4.1	General	15
4.2	Simple Request/Reply	15
4.3	Request/Reply Using an ESB	16
4.4	Events	16
4.5	Transactions	17
4.6	Callback	18
4.7	Adapters	19
4.8	Complex Messaging	20
4.9	Orchestration	20
4.10	Application-Level Use Cases	21
5	Integration Patterns	22
5.1	General	22
5.2	Client and Server Perspectives	22
5.2.1	General	22
5.2.2	Basic Web Service Pattern	22
5.2.3	Basic JMS Request/Reply Pattern	23
5.2.4	Event Listeners	24
5.2.5	Asynchronous Request/Reply Pattern	25
5.3	Bus Perspective	26

5.3.1	General	26
5.3.2	ESB Messaging Pattern Using JMS	27
5.3.3	ESB Messaging Patterns Using Web Service Request	28
5.3.4	ESB Request Handling to Web Service	28
5.3.5	ESB Request Handling via Adapter	29
5.3.6	Custom Integration Patterns	30
6	Message Organization	32
6.1	General	32
6.2	IEC 61968 Messages	32
6.2.1	General	32
6.2.2	Verbs	33
6.2.3	Nouns	34
6.2.4	Payloads	35
6.3	Common Message Envelope	36
6.3.1	General	36
6.3.2	Message Header Structure	37
6.3.3	Request Message Structures	40
6.3.4	Response Message Structures	43
6.3.5	Event Message Structures	48
6.3.6	Fault Message Structures	49
6.4	Payload Structures	49
6.5	Strongly-Typed Payloads	52
6.6	SOAP Message Envelope	53
6.7	Request Processing	54
6.8	Event Processing	54
6.9	Message Correlation	55
6.10	Complex Transaction Processing Using OperationSet	56
6.10.1	OperationSet Element	57
6.10.2	Patterns	59
6.10.3	OperationSet Example	61
6.11	Common Security Implementation	63
6.12	Representation of Time	63
6.13	Other Conventions and Best Practices	64
6.14	Technical Interoperability	64
6.15	Service Level Agreements	64
6.16	Auditing, Monitoring and Management	64
7	Payload Specifications	65
8	Interface Specifications	69
8.1	General	69
8.2	Application-Level Specifications	69
8.3	Web Service Interfaces	70
8.3.1	General	70
8.3.2	WSDL Structure	71
8.3.3	Document Style SOAP Binding	71
8.4	JMS	72
8.4.1	General	72
8.4.2	Topic and Queue Naming	72
8.4.3	JMS Message Fields	72

9	Security	74
10	Version Control	75
	Annex A (Normative): XML Schema for Common Message Envelope	77
	Annex B (Normative): Verbs.....	86
	Annex C (Normative): Procedure for Strongly Typed WSDL Generation	88
	C.1 General	88
	C.2 Service Naming	88
	C.3 WSDL Definition Steps	90
	C.4 Message Templates	92
	C.5 WSDL Templates	95
	Annex D (Normative): Generic WSDL	102
	Annex E (Informative): AMQP	104
	Annex F (Informative): Payload Compression Example	105

FIGURES

Figure 1 - Overview of Scope.....	8
Figure 2 - Simple Request/Reply	15
Figure 3 - Request/Reply Using Intermediaries	16
Figure 4 – Events	17
Figure 5 - Point-to-Point (One Way) Pattern.....	17
Figure 6 – Transaction Example	18
Figure 7 – Callbacks	19
Figure 8 - Use of Adapters	19
Figure 9 - Complex Messaging.....	20
Figure 10 - Application-Level Use Case Example	21
Figure 11 - Basic Request/Reply using Web Services	22
Figure 12 - Basic Request/Reply using JMS.....	23
Figure 13- Event Listeners using JMS.....	25
Figure 14 - Aynchronous Request/Reply Pattern.....	26
Figure 15 - ESB Content-Based Routing	27
Figure 16 - ESB with Smart Proxy and Content-Based Routing	28
Figure 17 - ESB with Proxies, Routers and Adapters	29
Figure 18 - ESB Integration to Non-Compliant Resources	30
Figure 19 - Messaging Between Clients, Servers and an ESB.....	33
Figure 20 – Verbs and their Usage.....	34
Figure 21 - Example Payload Schema	35
Figure 22 - Common Message Envelope.....	36
Figure 23 - Common Message Header Structure.....	39
Figure 24 - Request Message Structure	41
Figure 25 - XML for Example RequestMessage.....	42
Figure 26 - Example 'Get<Noun>' Profile	43
Figure 27 - Response Message Structure	44
Figure 28 - Reply Message States	45
Figure 29 - Error Structure	46
Figure 30 - XML for Example ResponseMessage	47
Figure 31 - XML Example of Payload Compression.....	47
Figure 32 - XML Example for Error ResponseMessage	48
Figure 33 - EventMessage Structure	48
Figure 34 - XML Example for EventMessage.....	48
Figure 35 - Fault Message Structure	49
Figure 36 - Message Payload Container - Generic	50
Figure 37 - Payload Usages.....	52
Figure 38 - Message Payload Container - Type Specific	52
Figure 39 - SOAP Envelope Example.....	54

Figure 40 - Message OperationSet Element.....	56
Figure 41 - OperationSet Details.....	58
Figure 42 - Transactional Request/Response (non-OperationSet).....	59
Figure 43 - Published Events (non-OperationSet)	60
Figure 44 – Transactional Request/Response (OperationSet)	60
Figure 45 - Published Event (OperationSet).....	61
Figure 46 - Information Models, Profiles and Messages	65
Figure 47 - Contextual Profile Design in CIMTool.....	66
Figure 48 - Example Message Payload Schema.....	67
Figure 49 - Example Payload XML Schema	68
Figure 50 - Example Message XML.....	68
Figure 51 - Example Complex Business Process	70
Figure 52 - Example Organization of Topics and Queues.....	73

INTRODUCTION

The purpose of this document is to define a set of implementation profiles for IEC 61968 using technologies commonly used for enterprise integration. More specifically, this document describes how message payloads defined by parts 3-9 of IEC 61968 are conveyed using web services and the Java Messaging System. Guidance is also provided with respect to the use of Enterprise service Bus (ESB) technologies. The goal is to provide details that would be sufficient to enable implementations of IEC 61968 to be interoperable. In addition, this document is intended to describe integration patterns and methodologies that can be leverages using current and future integration technologies.

The IEC 61968 series of standards is intended to facilitate *inter-application integration* as opposed to *intra-application integration*. Intra-application integration is aimed at programs in the same application system, usually communicating with each other using middleware that is embedded in their underlying runtime environment, and tends to be optimised for close, real-time, synchronous connections and interactive request/reply or conversation communication models. IEC 61968, by contrast, is intended to support the inter-application integration of a utility enterprise that needs to connect disparate applications that are already built or new (legacy or purchased applications), each supported by dissimilar runtime environments. Therefore, these interface standards are relevant to loosely coupled applications with more heterogeneity in languages, operating systems, protocols and management tools. This series of standards is intended to support applications that need to exchange data every few seconds, minutes, or hours rather than waiting for a nightly batch run. This series of standards, which are intended to be implemented with middleware services that exchange messages among applications, will complement, not replace utility data warehouses, database gateways, and operational stores.

This standard is based upon the EPRI Technical Report 1018795 and other contributed works.

Implementation Profiles for IEC 61968

1 Scope

1.1 Scope of IEC 61968

The IEC 61968 standard, taken as a whole, defines interfaces for the major elements of an interface architecture for distribution systems within a utility enterprise. Part 1: Interface Architecture and General Recommendations, identifies and establishes requirements for standard interfaces based on an Interface Reference Model (IRM). Parts 3 through 9 of IEC 61968 define interfaces relevant to each of the major business functions described by the Interface Reference Model.

As described in IEC 61968, there are a variety of distributed application components used by the utility to manage electrical distribution networks. These capabilities include monitoring and control of equipment for power delivery, management processes to ensure system reliability, voltage management, demand-side management, outage management, work management, automated mapping, meter reading, meter control and facilities management. This set of standards is limited to the definition of interfaces and is implementation independent. It provides for interoperability among different computer systems, platforms, and programming languages. Methods and technologies used to implement functionality conforming to these interfaces are considered outside of the scope of these standards; only the interface itself is specified in these standards.

1.2 Scope of this Document

This document is Part 100 of the IEC 61968 standard and specifies an implementation profile for the application of the other parts of 61968 using common integration technologies, including JMS and web services. Guidance is also provided with respect to the use of Enterprise Service Bus (ESB) technologies. This provides a means to derive interoperable implementations of IEC 61968 parts 3 through 9. At the same time, this standard can be leveraged beyond information exchanges defined by IEC 61968.

The following diagram attempts to provide an overview of scope, where IEC 61968 compliant messages are conveyed using web services or JMS. Through the use of an ESB integration layer, the initiator of an information exchange could use web services, which the receiver could use JMS, and vice versa. The integration layer also provides support for one to many information exchanges using publish/subscribe integration patterns and key functionality such as delivery guarantees.

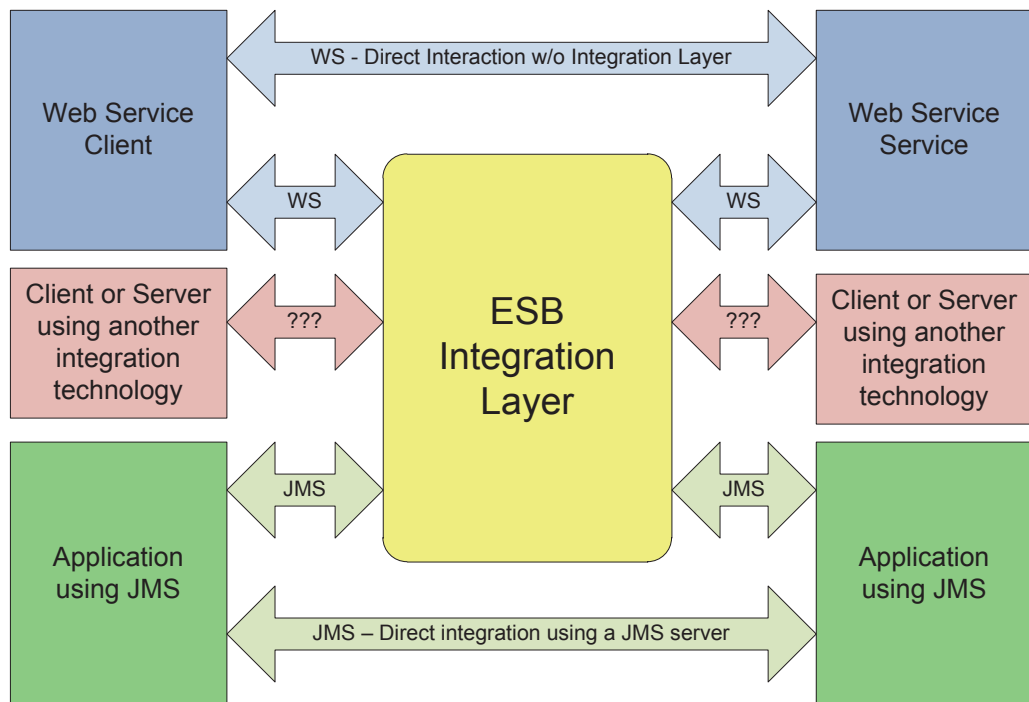


Figure 1 - Overview of Scope

The scope of this document specifically includes the following:

- Integration patterns that support IEC 61968 information exchanges
- Design of interfaces for use of strongly typed web services
- Design of interfaces for use of genericly typed web services
- Design of interfaces using JMS
- Definition of standard design artefacts and related templates
- Recognition that technologies other than JMS and web services may be used for integration leveraging this standard (with some specific examples and associated recommendations described in appendices)

This profile can also be applied to integration problems outside the scope of IEC 61968.

It is important to note that other implementation profiles can potentially be defined for IEC 61968, and that this is not intended to be the only possible implementation profile. In addition, this profile can be adapted to meet specific needs of specific integration projects.

It is also not within the scope of this document to prescribe those implementation details as required for security.

2 Normative References

The following documents contain provisions, which through reference in this text, constitute provisions of this International Standard. For dated references, only the editions cited apply. For undated references, the latest edition of the referenced document (including any amendments) applies. Members of IEC and ISO maintain registers of currently valid International Standards.

- IEC 60050-300, *International Electrotechnical Vocabulary (IEV) - Electrical and electronic measurements and measuring instruments - Part 311: General terms relating to measurements - Part 312: General terms relating to electrical measurements - Part 313: Types of electrical measuring instruments - Part 314: Specific terms according to the type of instrument*
- IEC 61970-301 *Energy Management System Application Program Interfaces – Part 301 Common Information Model Base.*
- IEC 61968-11 *Common Information Model (CIM) Extensions for Distribution*
- IEC 61968-1 *System Interfaces for Distribution Management – Interface Architecture and General Recommendations*
- IEC 61968-2 *Glossary*
- IETF RFC 1738: *Univeral Resource Locators*
- IETF RFC 2616: *Hyper-Text Transport Protocol 1.1*
- IETF RFC 4122: *A Universally Unique Identifier (UUID) URN Namespace*
- ISO-8601: *Representation of dates and times*
- *Extensible Markup Language (XML):* <http://www.w3.org/TR/REC-xml>
- *XML Schema:* <http://www.w3.org/XML/Schema>
- *Simple Object Access Protocol (SOAP) 1.2:* <http://www.w3.org/TR/soap12-part1/>
- *WS-I Basic Profile Version 1.0:* <http://www.oasis.org>
- *JMS API: JSR-914 Java Message Service (JMS) API* IEC 61970-552 *CIM XML Model Exchange Format*
- IEC 62361-100: *Naming and Design Rules for CIM Profiles to XML Schema Mapping*¹

¹ under consideration

3 Terms, definitions and abbreviations

3.1 Terms and definitions

3.1.1 General

For the purposes of this specification, the terms and definitions given in IEC 60050-300, IEC 61968-2, IEC 62051, IEC 62055-31 and the following terms apply.

Where there is a difference between the definitions in this standard and those contained in other referenced IEC standards, then those defined in 61968-2 shall take precedence over the others listed, and those defined in this document shall take precedence over those defined in 61968-2. This clause includes some definitions taken from Wikipedia.

3.1.2 Enterprise Service Bus

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, usually based on recognized standards, which provide foundational services for more complex architectures via an event-driven and standards-based messaging engine (the bus).

An ESB generally provides an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code. Contrary to the more classical enterprise application integration (EAI) approach of a monolithic stack in a hub and spoke architecture, the foundation of an enterprise service bus is built of base functions broken up into their constituent parts, with distributed deployment where needed, working in harmony as necessary.

An ESB does not implement a service-oriented architecture (SOA) but provides the features with which one may be implemented.

3.1.3 Java Messaging System (JMS)

The **Java Message Service (JMS)** API is a Java Message Oriented Middleware API for sending messages between two or more clients. JMS supports request/reply, publish/subscribe and point to point messaging patterns. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process as JSR 914. It is important to note that some ESB product vendors provide language bindings for JMS using C, C++ and/or C#, making the term JMS a misnomer. Where the wire protocol is different between different JMS implementations it is often trivial to bridge between different JMS implementations.

3.1.4 Service-Oriented Architecture (SOA)

Service Oriented Architecture (SOA) is a computer systems architectural style for creating and using business processes, packaged as *services*, throughout their lifecycle. SOA also defines and provisions the IT infrastructure to allow different applications to exchange data and participate in business processes. These functions are loosely coupled with the operating systems and programming languages underlying the applications. SOA separates functions into distinct units (services), which can be distributed over a network and can be combined and reused to create business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. SOA concepts are often seen as built upon and evolving from older concepts of distributed computing and modular programming.

3.1.5 Event-Driven Architecture (EDA)

Event-Driven Architecture (EDA) is a software architecture pattern that promotes the production, detection and consumption of events. An event is any change of state of potential interest. Within an EDA, events are transmitted between loosely coupled software components and services, typically using publish/subscribe messaging patterns. EDA is complementary to SOA, to the extent that SOA 2.0 is also known as 'event-driven' SOA. EDA is fundamental to a variety of business intelligence patterns, including complex event processing patterns.

3.1.6 Simple Object Access Protocol (SOAP)

The Simple Object Access Protocol (SOAP) is a standard that defines the formatting of XML messages. SOAP serves as a foundation layer of the web services protocol stack. SOAP is also commonly used within JMS. Common transports for SOAP include HTTP, HTTPS and proprietary JMS transports. SOAP is also now sometimes referred to as 'Service-Oriented Architecture Protocol'. SOAP is a W3C Recommendation.

Two versions that are in common use include SOAP 1.1 and SOAP 1.2. New integrations or interfaces should use SOAP 1.2 when applicable.

3.1.7 Web Services (WS)

A Web Service is defined by the W3C as 'a software system designed to support interoperable Machine to Machine interaction over a network.' Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

The W3C Web service definition encompasses many different systems, but in common usage the term refers to clients and servers that communicate using XML messages that follow the SOAP standard. Common in both the field and the terminology is the assumption that there is also a machine readable description of the operations supported by the server written in the Web Services Description Language (WSDL). The latter is not a requirement of a SOAP *endpoint*, but it is a prerequisite for automated client-side code generation in many Java and .Net development tools.

Where web services have two primary styles, document-centric and RPC-centric, the focus of this specification is document-centric. This is supportive of SOA and the transport of IEC 61968 payloads. For increased interoperability, the document wrapped form is required.

It is important to note that web services do not readily support publish/subscribe messaging unless mechanisms such as WS-Eventing are used. However, this specification also describes an approach for the ESB to route messages asynchronously to configured subscribers.

This standard will discuss two approaches for use of web services, where WSDL operations are either generic or strongly typed.

3.1.8 Web Services Definition Language (WSDL)

Web Services Definition Language (WSDL) is an XML-based language that is used to describe web services. WSDL is often used in combination with SOAP and XML schema to provide web services over the internet (or an intranet). WSDL is a W3C Recommendation.

Two versions that are in common use include WSDL 1.1 and WSDL 2.0. WSDL 2.0 better supports interoperability between Java and .Net implementations.

The WSDL templates provided by this document are based upon WSDL 1.1, and consequentially WSDL 1.1 is a requirement.

3.1.9 XML Schema (XSD)

XML Schema, published as a W3C recommendation in May 2001, is one of several XML schema languages. It was the first separate schema language for XML to achieve Recommendation status by the W3C.

Like all XML schema languages, XML Schema can be used to express a schema: a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema. However, unlike most other schema languages, XML Schema was also designed with the intent that determination of a document's validity would produce a collection of information adhering to specific data types. An XML Schema instance is an XML Schema Definition (XSD) and typically has the filename extension ".xsd". The language itself is sometimes informally referenced as XSD.

It is important to note that 61968 payload are defined using XML schemas. Those XML schemas provide normative specifications for application payloads that are conveyed using this standard.

3.1.10 Representational State Transfer (REST)

Representational State Transfer (REST) is an alternative to the use of SOAP-based web services. REST itself is not currently a standard, but instead an architectural style.

REST leverages HTTP, where each URL is a representation of some object. Interfaces can be defined in terms of XML payloads for requests and responses. A WSDL is not required for the use of REST. Additionally, the XML documents used for requests and responses do not need to be defined using an XSD, although it is common for the XML to be compliant to an XSD.

Within REST, an object can be retrieved (read) using an HTTP GET. Similarly, an HTTP POST is used to create an object, an HTTP PUT is used to modify an object and an HTTP DELETE is used to delete an object.

REST is discussed here for completeness purposes only as it may be encountered by an integration project. However, there are currently no specific recommendations for mappings by this standard. This may be a future annex.

3.1.11 Queue

A queue is a construct supported by many messaging products to provide reliable messaging with delivery guarantees. A standard API that includes queue-based messaging models is provided by JMS. AMQP also defines an open protocol for queue based messaging.

3.1.12 Topic

A topic is a construct supported by many messaging products to enable publish/subscribe messaging patterns where there may be potentially many consumers of a message that has been sent to a named topic by a publisher. Topics are commonly used as a destination for event messages. Topics are directly supported by JMS.

3.1.13 Message Destination

A message destination is the target address for a message, whether it be a request, response or an event message. When using JMS, the destination may be a topic or queue. When using HTTP mechanisms such as web services, the destination is specified as a URL.

3.1.14 Request

A request is a message sent from a client (or source) to a server (or target) where a response is expected. The request may be either a query (where data is returned from the target) or a transaction (where data is modified in the target). A request will use verbs such as 'get', 'create', 'change', 'delete', 'cancel', 'close' or 'execute'.

3.1.15 Response

A response is a message sent as a consequence of a request, typically from the target of the request to the source of the request. Response messages are synonymous with 'reply' messages. A response message will use a 'reply' verb.

3.1.16 Query

A query is a type of request where the target is expected to return information to the source of the request. The request message for a query will use the 'get' verb. This will typically be implemented using a request/reply pattern.

3.1.17 Transaction

A transaction is a type of request where the target typically will modify information that it manages. A transaction request will use verbs such as 'create', 'change', 'delete', 'cancel', 'close' or 'execute'. This may be implemented using a variety of patterns. If a pattern other than request/reply is used, there should be a delivery guarantee provided by the transport.

3.1.18 Event

The term 'Event' is significantly overloaded, and can have different meanings in different contexts. The most general definition of an event is 'something that happens at a given place and time', or 'a change of state of potential interest'. As a consequence of an event, 'something' may generate an 'event message' to report the fact that a certain type of event occurred at a given time. Event messages are therefore asynchronous in nature and are typically published to potentially interested subscribers using topic-based messaging.

Event messages are one type of asynchronous message. It may be common for an application to generate an event when a condition of interest is detected, or a transaction has been processed. There are several integration patterns that are related to the support of events.

Events are typically published asynchronously as event messages by a system or application in order to report conditions of interest. In some cases a system or application may internally identify a condition of interest, but in other cases the condition may be detected by an external source such as a device. Event messages will use past tense verb such as 'created', 'changed', 'deleted', 'closed', 'canceled' or 'executed'.

3.2 Abbreviations

The following terms and abbreviations are used within this document:

API	Application Programming Interface
AMQP	Advanced Message Queue Protocol
CIM	Common Information Model
CME	Common Message Envelope
CRUD	Create, Read, Update, Delete
EDA	Event Driven Architecture

ESB	Enterprise Service Bus
IEC	International Electrotechnical Commission
IETF RFC	Internet Engineering Task Force Request For Comments
ISO	International Standards organization
JEE	Java Enterprise Edition
JMS	Java Message Service
JSR	Java Specification Request
mRID	CIM master resource identifier
OASIS	Organization for the Advancement of Structured Information Standards
RDF	Resource Description Framework
REST	REpresentational State Transfer
RFC	Request for Comments
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secured Socket Layer
TLS	Transport Layer Security
UML	Unified Modelling Language
URL	Uniform Resource Locators
UUID	Universal Unique Identifier
W3C	World-Wide Web Consortium
WS	Web Services
WS-*	Web Services standards
WS-I	Web Services Interoperability
WSDL	Web Services Definition Language
XML	eXtensible Markup Language
XSD	XML Schema
XSL	XML Stylesheet Language

4 Use Cases

4.1 General

The purpose of this clause is to describe several use cases related to the interactions between components within a set of systems cooperating to support a set of business processes. It is important to note that the use cases presented are from the perspective of the integration of systems, as opposed to end use application-level use cases. The actors for the use cases described in this clause include the following:

- Client
- Server
- ESB
- Adapter
- Subscriber (an Event Listener)

Three key terms related to messaging are request, reply and event. Within the terminology of IEC 61968, these are reflected in terms of the verbs used to define specific information flows.

Central to the use cases is the assumption that a variety of integration technologies may be used, where the focus of this standard is JMS and web services.

4.2 Simple Request/Reply

The first use case is a simple request/reply between a client and server. This is synonymous with request/response. The initiator of the request is the client, where the requested is processed by the server. The first view of this is the simple view without the use of an ESB. This use case involves one of two cases:

1. A client making a query request to a server, where the server will return a set of objects to the client based upon some filter criteria
2. A client making a transaction request to a server, where a set of objects will be created or modified in some way

Both cases are illustrated by the following diagram.

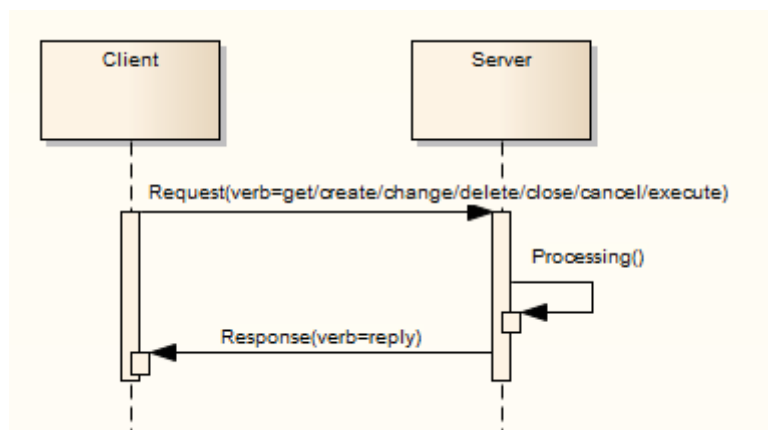


Figure 2 - Simple Request/Reply

In terms of IEC 61968, requests use the verbs such as *get*, *create*, *change*, *delete*, *close*, *cancel* or *execute*.

4.3 Request/Reply Using an ESB

The simple request/reply use case can also be extended to leverage an ESB. Within the ESB many actions can be taken by intermediaries as needed to facilitate integration and the decoupling of components, such as transformations and routing.

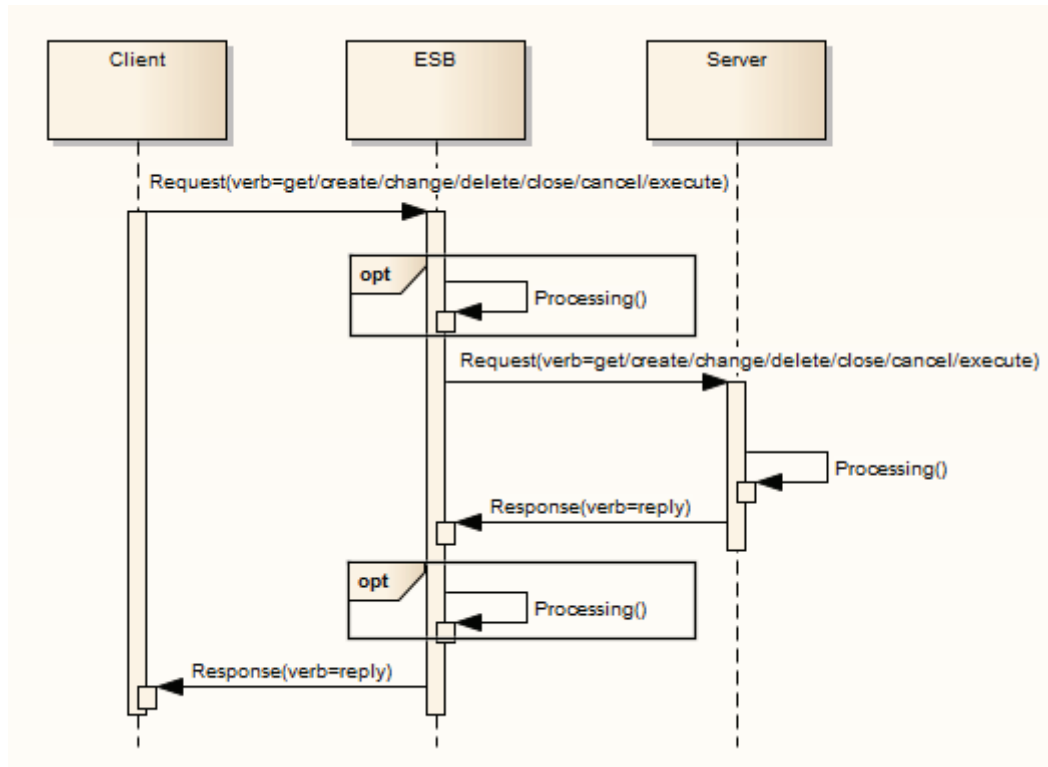


Figure 3 - Request/Reply Using Intermediaries

A key aspect of this use case is the decoupling of the client from the server, so that the client does not have to know the exact location of the server or conform to the exact interface used by the server. The routing and mapping can take place in the integration layer.

It is also recommended that ESB intermediaries be stateless. The use of stateless intermediaries simplifies the implementation of load balancing and high availability.

4.4 Events

There is often the need for client processes to be informed of an event of potential interest. Many client processes can listen (subscribe) for events. One example of this is EndDeviceEvents messages that may be published by a metering system. Other examples include events that report the execution of a control or transaction, such as the creation or update of a work order.

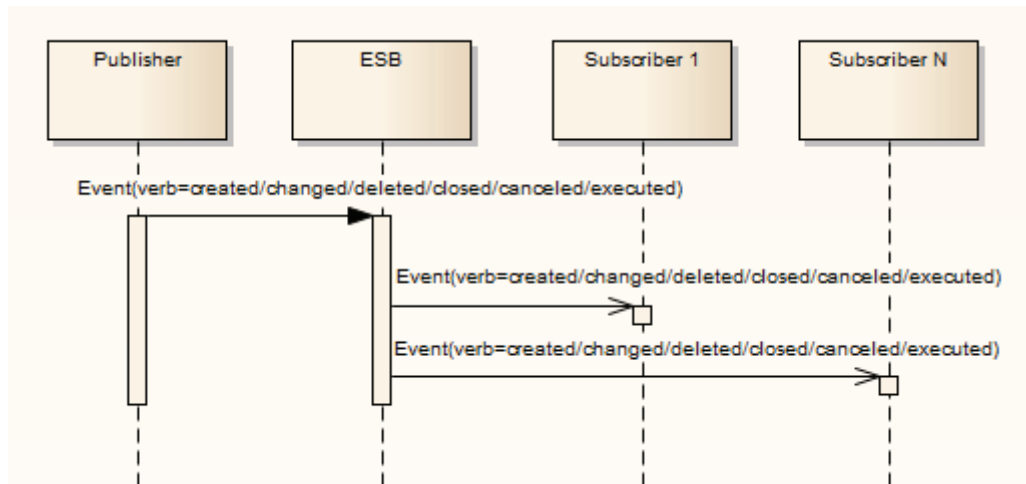


Figure 4 – Events

In terms of IEC 61968, events use the ‘past tense’ verbs *created*, *changed*, *deleted*, *canceled*, *closed* or *executed*.

This is an example of a one way pattern, where the sender does not expect a response at the application level, although the transport used may allow for lower level acknowledgements in order to facilitate delivery guarantees when needed.

Listeners that use web services will need to have an exposed interface at a URL that is known by an intermediary so that the events can be appropriately distributed. Rules must also be defined for retry processing where guaranteed delivery is required. The implications of message ordering also need to be considered.

4.5 Transactions

The use case for a transaction is typically a combination of a request/reply exchange between a client and a server, with a consequential publication of events. An important aspect is that the clients and services may use different transport mechanisms, where a client may use web services but a client may use JMS. An intermediary within the ESB can provide the necessary routing functions.

Transactions are commonly implemented using a request/reply pattern, but they can also be implemented using a point-to-point or one way pattern.

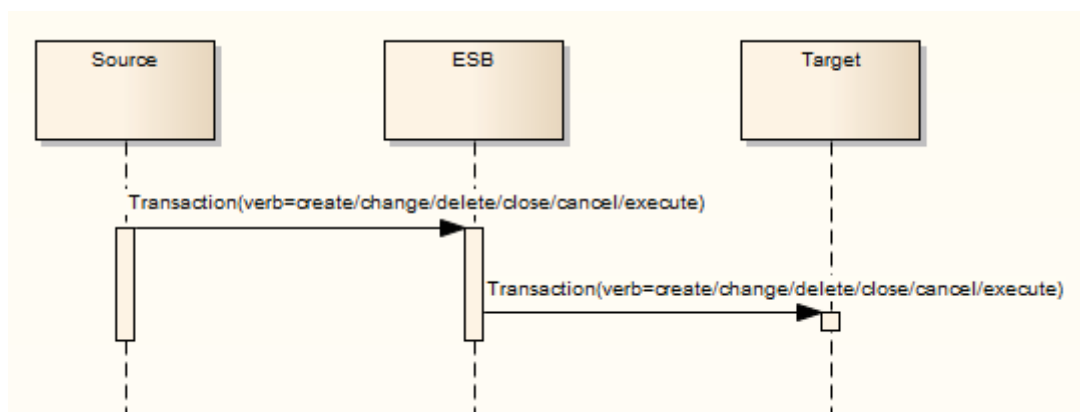


Figure 5 - Point-to-Point (One Way) Pattern

The one way pattern is used in cases where a system of record wants to forward a transaction to a peer system that also must apply the transaction. If errors are detected by the target, they must be logged for resolution within the target system. The following figure provides an example where a transaction will result in events that get propagated to interest subscribers.

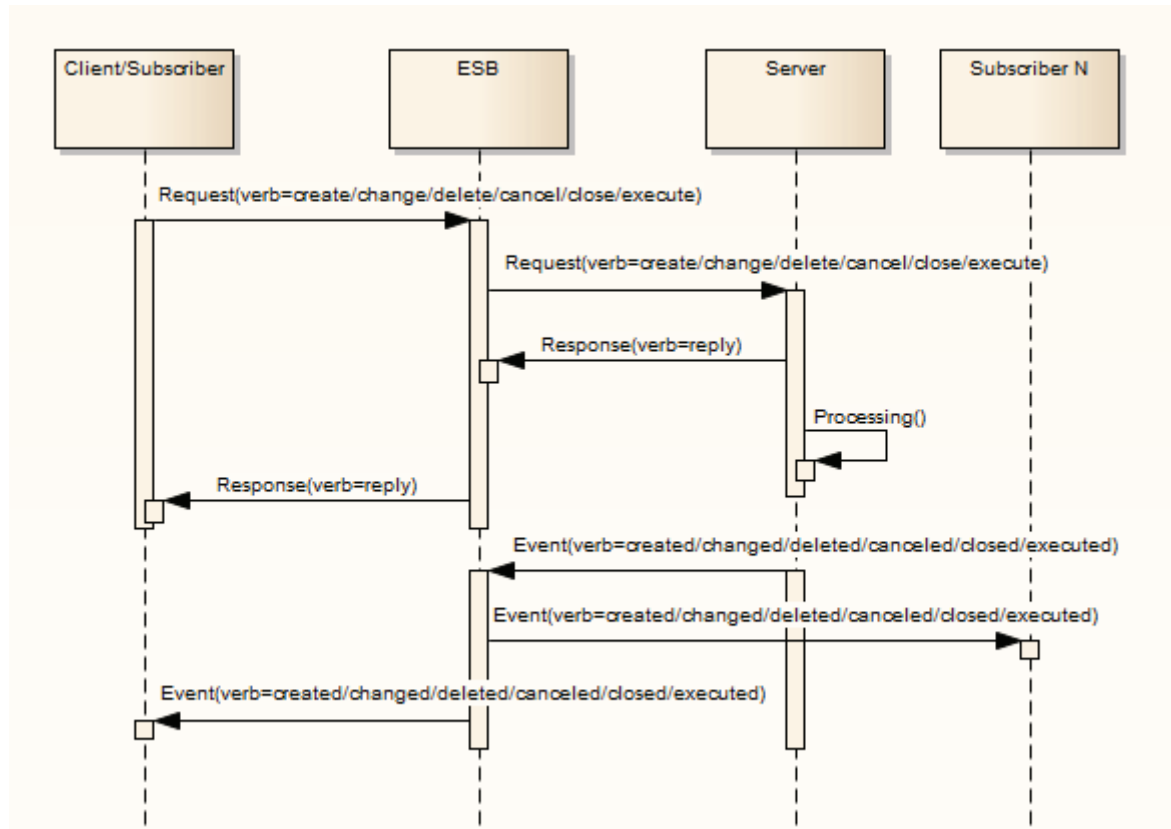


Figure 6 – Transaction Example

In terms of IEC 61968, requests related to transactions use the verbs create, change, delete, cancel, close, and execute. Verbs related to events use past tense. The use of the execute verb implies a complex transaction and the use of the Payload.OperationSet element, as described in section 6.9.

4.6 Callback

A 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. In this case, the client sends an initial request to server. Once the server receives the message, it returns a response message back. At this point the initial message transaction is completed and client application is freed to perform other processing. Once the server has completed processing, it then invokes the final request/response sequence with a request message. The whole call-back process is completed after the client (of the initial request) replies to the final request. This is illustrated in the following sequence diagram.

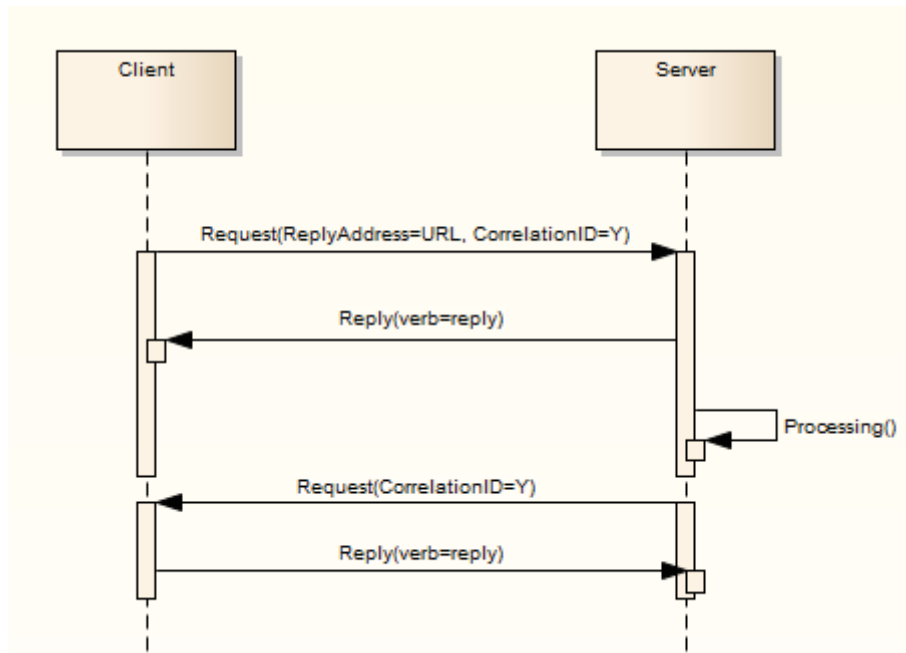


Figure 7 – Callbacks

In the call back process, the client has to inform the server of the location for the final response, often a URL called a callback address. This piece of information is included in the initial request message. Section 6.3.2 discusses specific message elements that are used to control this type of dialog.

4.7 Adapters

There are cases where an application (client or server) cannot directly connect to an ESB. In these cases, an adapter can be used to handle the 'impedance mismatch' between the application and the ESB. In some cases the application may simply be a database or file directory. In the following diagram an adapter is used to connect a server to the ESB.

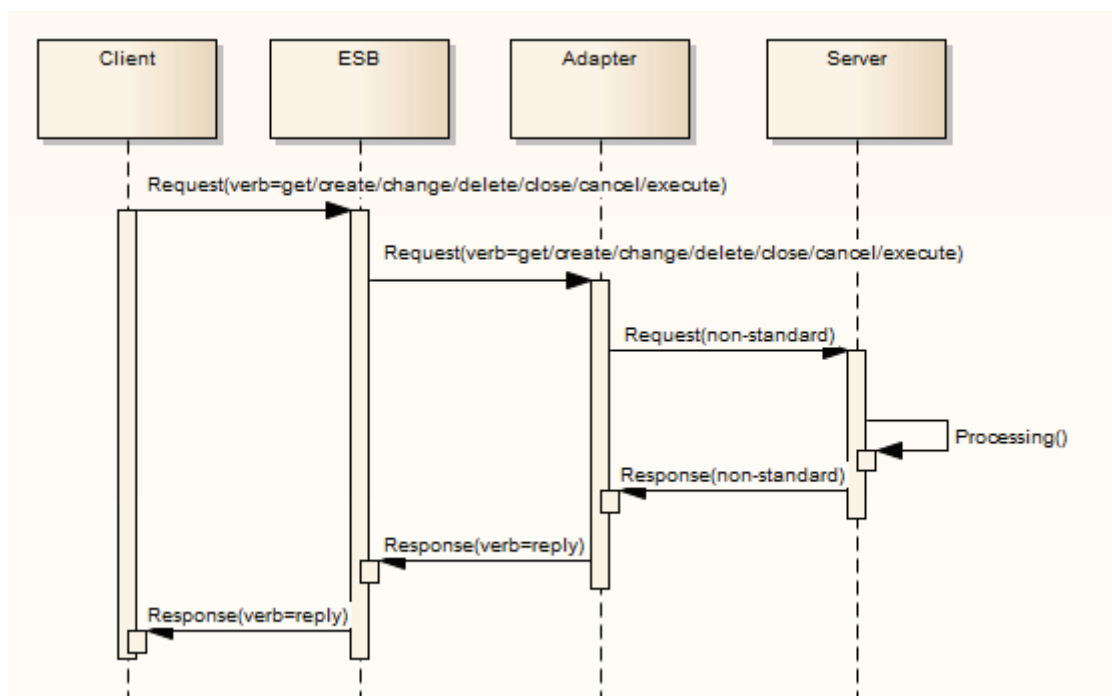


Figure 8 - Use of Adapters

The adapter may perform a variety of functions, and may take actions on behalf of the application such as generation of events that are consequential to the success of a transaction. The most common activity is to convert data between an application model and an enterprise canonical model.

4.8 Complex Messaging

There are some use cases that may require more complex messaging patterns. For example, there may be cases where a request may involve potentially many asynchronous replies to return information as results become available. The results can be returned to the client in the form of asynchronous events.

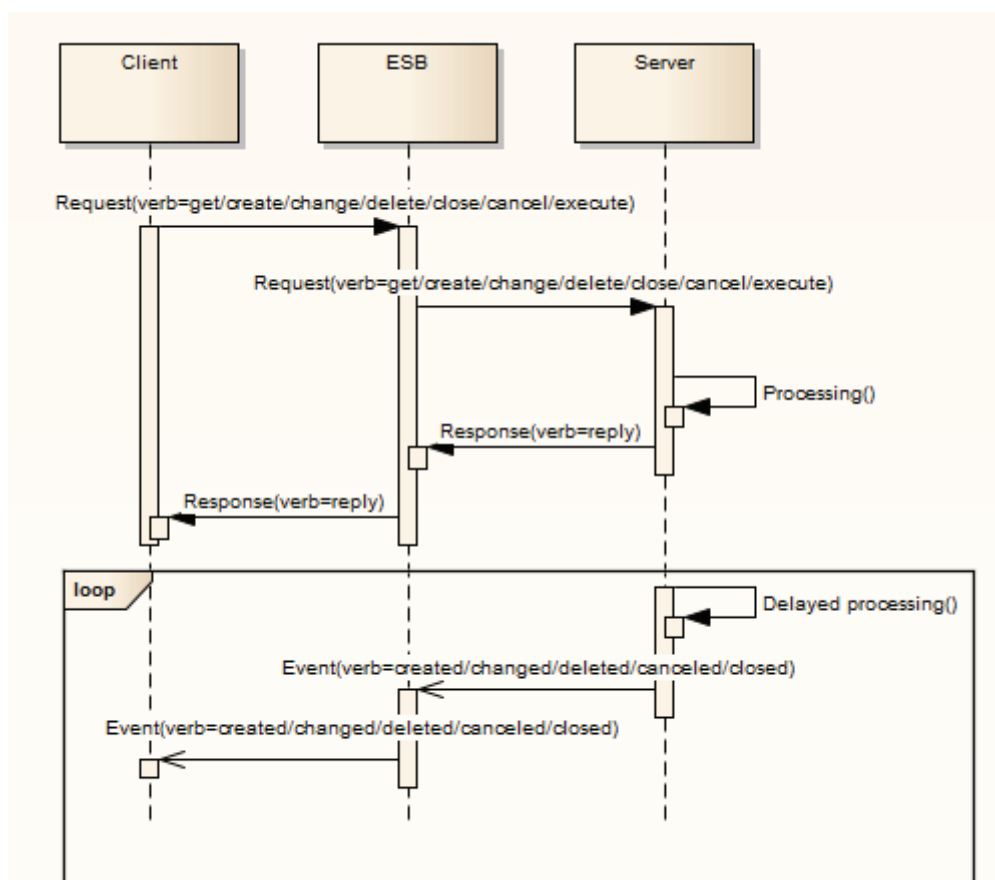


Figure 9 - Complex Messaging

The above use case is seen in applications such as metering, where it may take significant time to obtain results as in the case of a disconnect or load control, where an EndDeviceControls message might be issued, but the results may be reported after some delay using an EndDeviceEvents message.

It is important to note that this is a variation of the callback pattern. The initial request could either be a query, where results are returned asynchronously, or a transaction where one or more events could be generated as a consequence.

4.9 Orchestration

Use cases related to the orchestration or choreography of a business process as well as short or long running distributed transactions are outside the scope of this specification. The integration approaches described by this document can be leveraged by more complex integrations that address those needs.

4.10 Application-Level Use Cases

The use cases described in the previous subclauses can be applied to end use application-level use cases. In the following example application-level use case, messages are defined using IEC 61968 verbs and nouns, in the form '<verb>(<Noun>)'. Specific (or representative) types of systems are also identified as opposed to more generic types of actors.

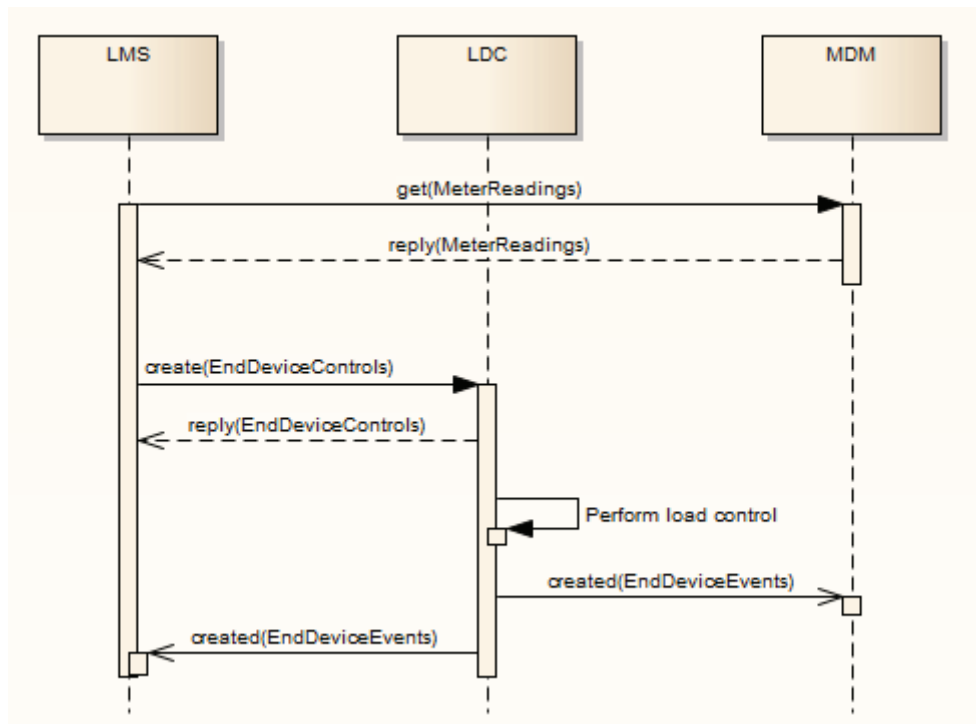


Figure 10 - Application-Level Use Case Example

5 Integration Patterns

5.1 General

This document recognizes a set of basic service integration patterns that support the previously described use cases. It also allows for more complex integration patterns through the use of intermediaries within an enterprise server bus (ESB).

5.2 Client and Server Perspectives

5.2.1 General

From the perspective clients and servers there are several basic integration patterns described within this document. These patterns include, but are certainly not limited to:

- 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
- 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

From the perspective of either a client alone or a server alone, whether or not the client is communicating with the server directly, or through intermediaries is irrelevant so long as they have chosen to use the same transport mechanism. The value of an ESB is that an integration architecture can be provided where a client has a choice of using web services, topics or queues, and each target service can also choose to use web services, topics or queues independently from the choices of any client.

5.2.2 Basic Web Service Pattern

The following diagram illustrates a basic request/reply pattern using web services.

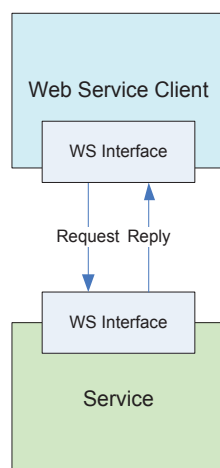


Figure 11 - Basic Request/Reply using Web Services

In this pattern the client issues a request to a web service interface exposed by some service. This interface is defined using a WSDL. However, this interaction pattern may also be realized using REST. The client should expect one of several outcomes:

- The request is successfully processed, where a reply message is returned in a timely manner (Result=OK)
- The request is accepted, but results in a reply message that returns an application level error code (Result=FAILED)
- The request is accepted, but results in a reply message with a partial set of results (Result=PARTIAL)
- The request results in a fault being returned to the client
- After sending the request, no reply or fault is returned in a timely manner

It is also to note that there may be varying levels of security that may be required for the implementation. The extreme case is where the client is using a public network to communicate with the service, where authentication, authorization, encryption and signing may be important.

In the case where a reply is not required, this is called a 'one way' pattern. However care should be taken with respect to any required delivery guarantees.

5.2.3 Basic JMS Request/Reply Pattern

The following diagram describes a basic request/reply pattern where the client sends a JMS message to a topic (or queue). In this pattern the reply message is optional, where there may be some cases where a reply message is never sent.

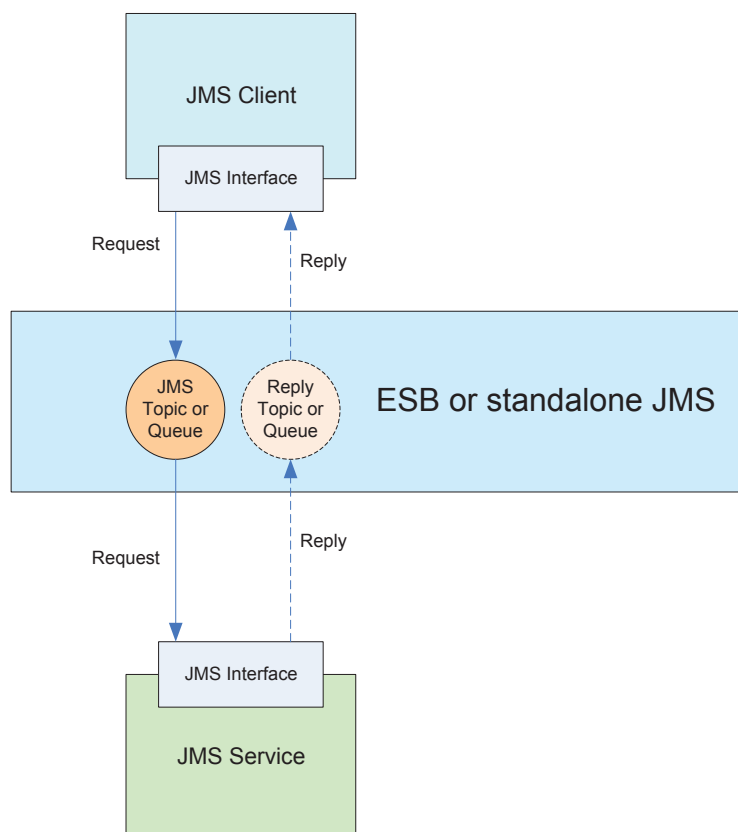


Figure 12 - Basic Request/Reply using JMS

A service listening on a message destination (topic or queue) consumes the request message and issues a reply to the client. The message destination is managed by the JMS implementation (typically part of an ESB). The client may consume the reply message synchronously or asynchronously, with the decision being left solely to the discretion of the client implementation.

Where this document refers to a JMS 'message destination', implementations may use either topics or queues for request/reply messaging. When using topics, a service would typically use a durable subscription in cases where request messages must not be lost.

If the client initiating the request expects a reply, it can expect one of the following results:

- The request message is successfully sent to a topic (or queue), where a reply message (correlated to the request using an ID) is returned in a timely manner (Result=OK)
- The request message is sent, but results in a reply message that returns an application level error code (Result=FAILED)
- The request is accepted, but results in a reply message with a partial set of results (Result=PARTIAL)
- The attempt to send a request message to the topic (or queue) fails
- After sending the request message, no reply message is ever received (which may or may not be normal behaviour depending upon the service)

Reply messages are sent using topics or queues as appropriate, where the specific topics may be statically or dynamically defined. For simplicity subsequent diagrams will not explicitly identify reply topics or queues. The delivery guarantees offered by JMS implementations provide the option for clients to making transaction requests to not require replies, where this is sometimes referred to as a one-way pattern.

JMS is typically used within a secure, private enterprise network. However there may be isolated cases where this is not the case and security is a significant concern. JMS can also be readily configured to use SSL/TLS and/or use client authentication.

This pattern actually does not require a full ESB implementation, where only a JMS implementation is actually needed. It is also important to note that JMS implementations from different vendors are typically not interoperable, and a 'bridge' may be required in cases where clients can not use a common JMS implementation.

5.2.4 Event Listeners

Another integration pattern is that of a process that listens for events that may be published. There are many cases where a service may publish event messages that are of potential interest to many other processes.

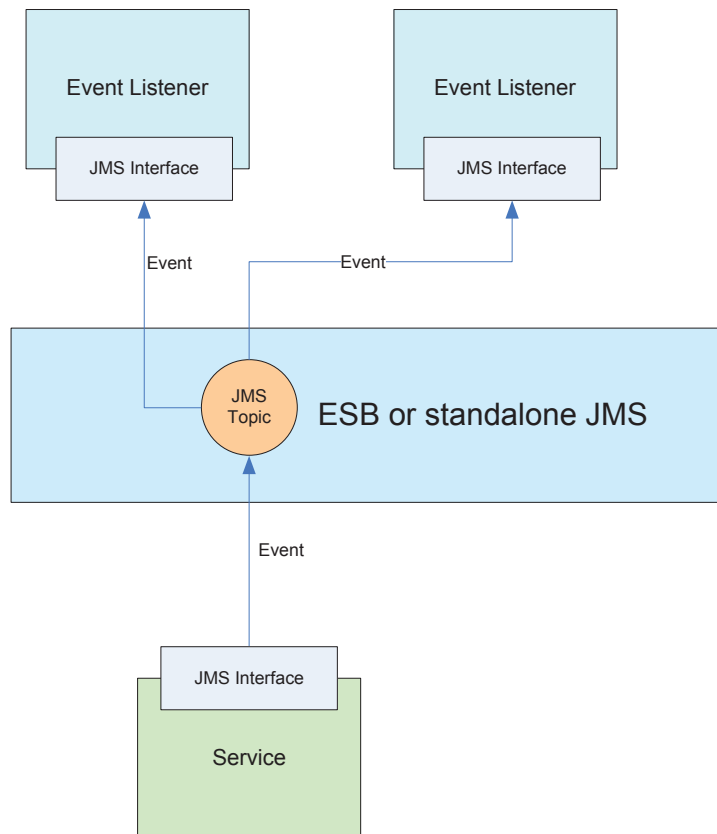


Figure 13- Event Listeners using JMS

Events should be published on topics (as opposed to queues), as there will typically be many consumers of events. The listener is responsible to subscribe to one or more JMS topics of potential interest. Some listeners may choose to use a durable subscription in cases where events must not be lost. Event messages are sent and consumer asynchronously. There is no acknowledgement message of any kind returned to the service.

There may be issues related to the number of topics (or queues) and the fan out (i.e. number of event listeners) that can be supported using a given JMS implementation.

It is also possible to extend the architecture described by this document to leverage the use of WS-Eventing for the publication and subscription of events by web service clients. There is currently an open source of WS-Eventing through the Apache project. However, this document will not focus on specific aspects of the implementation and use of WS-Eventing. This document also provides another example of routing events using web services.

5.2.5 Asynchronous Request/Reply Pattern

The asynchronous request/reply allows multiple replies to be returned to a requesting client asynchronously. This is a complex integration pattern that is slightly more complex to implement using web services than for JMS.

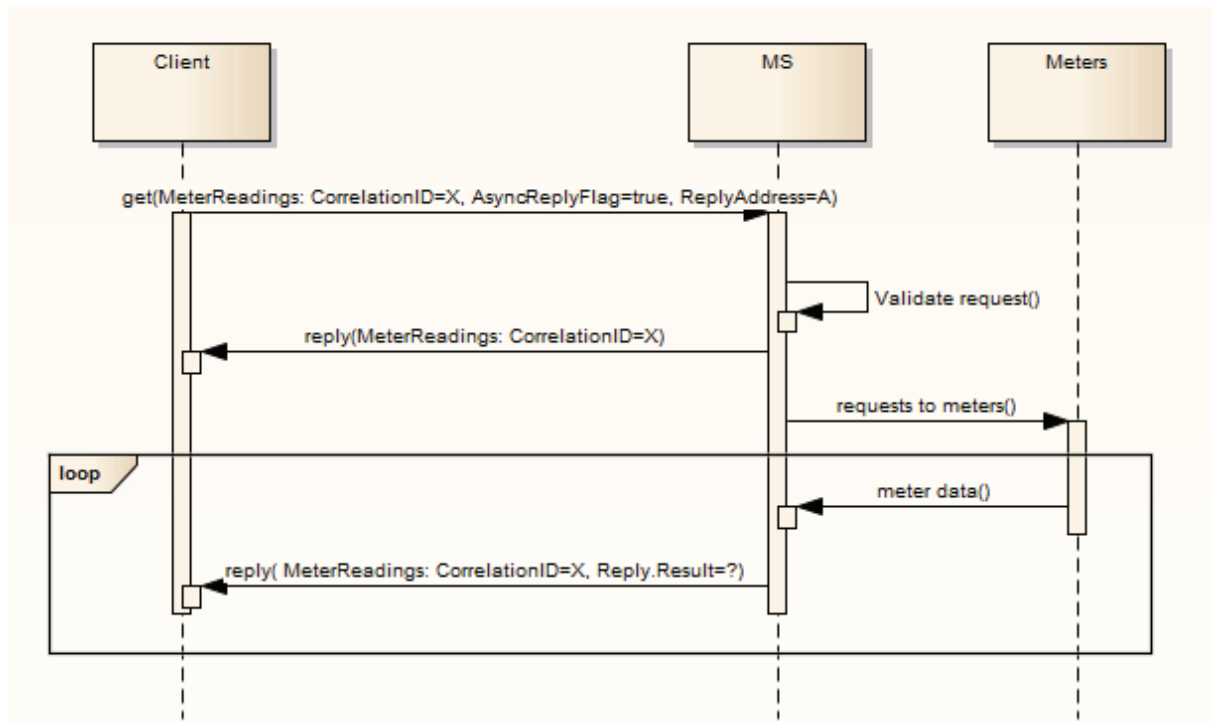


Figure 14 - Asynchronous Request/Reply Pattern

In this pattern, a client makes a request to a service. Where the service may not be able to process and/or respond with the desired information immediately, the service responds immediately with a trivial acknowledgement of the request. After processing is able to occur and/or the desired information is obtained, the service can provide one or more asynchronous replies to the client using a specified URL, topic or queue. The key elements (as described in section 6) used to control this exchange include:

- Header.CorrelationID is used to logically link all messages together. Where a CorrelationID is provided by the client on the initial request, the server must include it on all related response and event messages.
- Header.AsyncReplyFlag is set to 'true' on the initial request.
- Header.ReplyAddress is set on the initial request to identify the destination where replies should be set.
- Reply.Result is used on responses, where 'PARTIAL' indicates that more responses may be expected, or 'OK' indicates that processing is complete and no more responses should be expected.

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.

5.3 Bus Perspective

5.3.1 General

Clients and servers can either communicate directly, or through intermediaries. The enterprise service bus (ESB) is used for the management of intermediaries. The use of an ESB provides for many variations in communication patterns. However, the client still sees the ESB as being no different than a server. Similarly, the server sees the ESB as being no different than any other client.

This is important in that:

- No knowledge of the ESB is imposed upon any client or server implementation of an IEC 61968 interface
- There is no requirement for an ESB product placed upon any IEC 61968 interface, except that a JMS server is needed where JMS messages are to be used
- A project implementation using IEC 61968 can use an ESB and freely implement integration patterns that are appropriate for the project and the associated integrations

Given that an ESB is not required for the implementation of an IEC 61968 interface, the remainder of this subclause describes recommendations only, and is provided as informative material.

5.3.2 ESB Messaging Pattern Using JMS

The basic ESB messaging pattern using JMS introduces the option for routing of requests. This serves to further decouple the client and server, where the bus (through use of routing logic, often referred to as a 'Content-Based Router' integration pattern [EIP]) can make decisions related to the handling of the request. This is described in the following diagram.

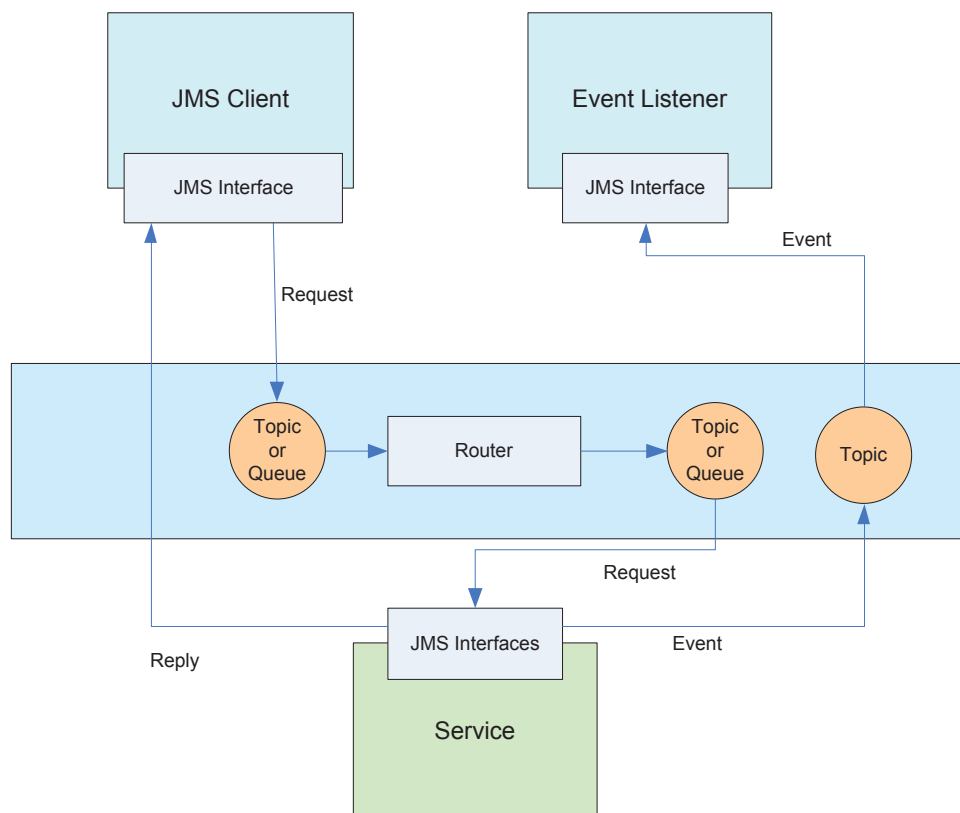


Figure 15 - ESB Content-Based Routing

When a client issues a request to a topic (or queue), a router on the bus can decide to forward the message to another topic (or queue). The decisions by the router may take into account any of the following:

- Contents of a message header
- Contents of a message payload (although this should be avoided)

- The status of a destination service instance
- The need to balance load

It is important to note that one advantage of the loosely coupled approach described by this document is that routing components are not tied to messages of specific payload types. The router can be configured using XPath expressions to identify message content to determine actual routing.

5.3.3 ESB Messaging Patterns Using Web Service Request

The following diagram extends the previously describe pattern to permit a request to be initiated by a web service client as well as a JMS client.

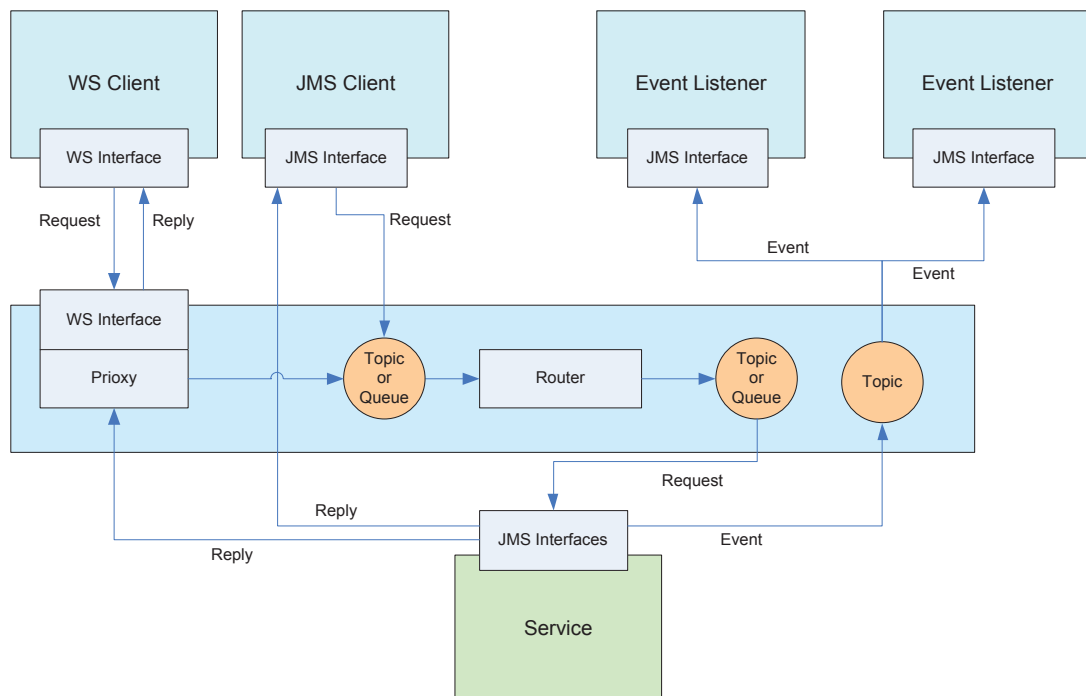


Figure 16 - ESB with Smart Proxy and Content-Based Routing

A proxy (sometimes referred to as a ‘Smart Proxy’ integration pattern [EIP]) component is implemented on the ESB to expose a web service (as defined by a WSDL). The Smart Proxy can make decisions with respect to dispatching of requests and correlation of responses. The message conveyed through the WSDL is simply converted to a JMS message and is then routed as appropriate.

5.3.4 ESB Request Handling to Web Service

The following diagram extends the previous pattern to allow for a service to expose its interface as a web service with an appropriately defined WSDL.