Creating A Single Global Electronic Market

ebXML Transport, Routing & Packaging Overview and Requirements

Working Draft 26-May-2000

This version:
ebXML TR&P Overview & Requirements (v 0-96).doc
Latest version:
(URL to latest version)
Previous version:
(URL to previous version)

Editor:
David Burdett <david.burdett@commerceone.com>
Authors:
David Burdett
Contributors:
ebXML Transport, Routing & Packaging team members.
See Acknowledgements

Abstract

This paper provides an overview of the ebXML Transport Routing and Packaging and a description of the requirements that have been identified.

It describes:

- an overview and description of the scope of the group's work
- the objectives of the group
- a draft diagram that outlines the relationship of the group to other groups within ebXML
- the requirements for Transport, Routing and Packaging
- a definition of the terms used in the description of the requirements, and
- some examples of how the different sequences in which message can be exchanged

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Status of this Document

This document is a draft for Public Comment. The document represents work in progress and no reliance should be made. It has been updated to reflect the results of the ebXML Transport, Routing and Packaging project team's meeting during the ebXML conference in Brussels in early May 2000. Most changes have been made to section 4. Requirements.

Table of Contents

1 Introduction 2

2 Objectives 3

3 Relationships with other ebXML activities 3

4 Requirements 7

4.1 Envelope and headers for business documents 7

4.2 Reliable Messaging and Error Handling 8

4.3 Message Routing 9

4.4 Security Requirements 9

4.5 Audit Trails 10

4.6 Quality of service 10

4.7 Platform Independent Interoperability 11

4.8 Restart and recovery 11

4.9 Protocol Extensibility 12

5 Definitions 12

5.1 Documents, Parties, Messages and Document Exchanges 12

5.2 Services and Message Sets 17

5.3 Miscellaneous 20

6 Examples of Document Exchanges 20

7 References 23

8 Acknowledgements 23

1 Introduction

In outline the working group will develop deliverables that:

1) provide an envelope and header for routing of message content
2) define template sequences for the exchange of messages
3) provide support for payloads of any type of digital data
4) adopt security protocols that enable:
    a) non repudiation of sending of messages and acknowledgements
    b) privacy and integrity of communications between parties
    c) authentication of senders of messages
    d) control over access to services

5) support verifiable audit trails

6) provide mechanisms for reporting on errors or other problems

7) support a messaging protocol for reliable message delivery

8) define the information required that describes how to interact with a service

9) provide a default method of usage that enables bootstrapping of services

2 Objectives

The objectives of the working group are:

1) to enable any party to carry out integrated eCommerce transaction with any other party anywhere in the world using their hardware and software vendor of choice

2) to persuade a wide variety of vendors to implement the approach

3) to not reinvent the wheel - re-use where possible

4) to enable existing "messaging" solutions to "bridge" to the ebXML solution

5) to scale from SMEs to large companies

6) to scale from low power to high end solutions

3 Relationships with other ebXML activities

This section contains a number of diagrams that explain the relationship between the Transport, Routing and Packaging Group and other activities within ebXML. Definitions of words or phrases in italics may be found in section 5 Definitions.

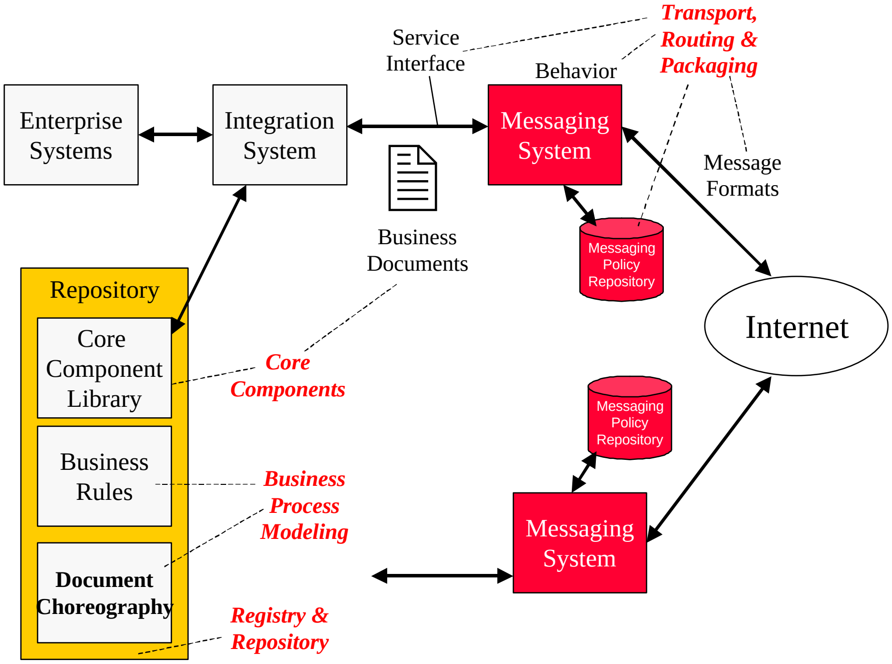


Figure 1 Relationship between ebXML activities

This diagram illustrates the relationship between the work of the Transport Routing and Packaging group (TR&P) and the other groups of ebXML. A more detailed description of the scope of the TR&P group is shown by the diagram below.

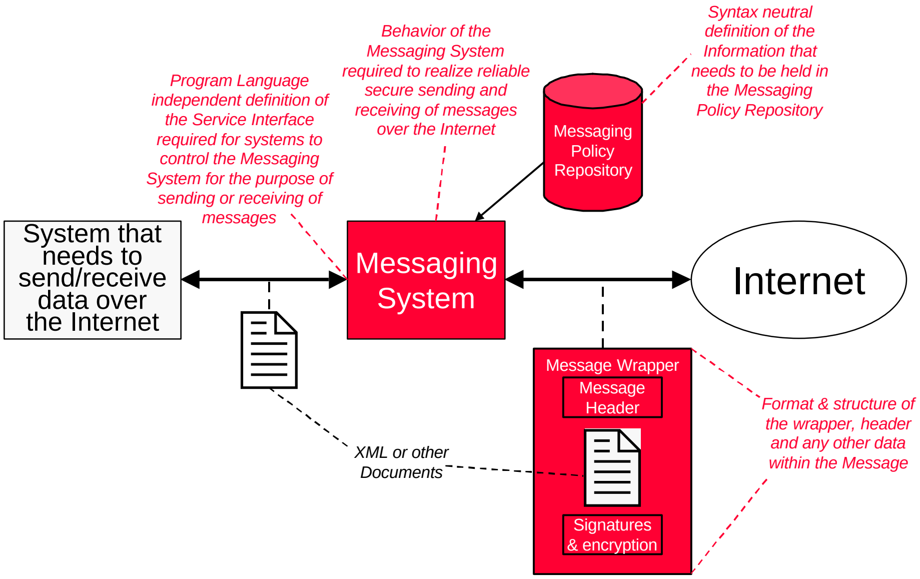

Program Language independent definition of the Service Interface required for systems to control the Messaging System for the purpose of sending or receiving of messages
Behavior of the Messaging System required to realize reliable secure sending and receiving of messages over the Internet
Syntax neutral definition of the Information that needs to be held in the Messaging Policy Repository
Messaging Policy Repository
System that needs to send/receive data over the Internet
Messaging System
Internet
Message Wrapper
Message Header
Signatures & encryption
XML or other Documents
Format & structure of the wrapper, header and any other data within the Message

Figure 2 Scope of Transport, Routing and Packaging Activities

The scope of the Transport, Routing and Packaging group is defined by the items colored red in the diagram above. All specifications will be produced initially in a syntax neutral and/or language independent way.

The intention is that representations of this information in specific languages or syntax are then separately developed. For example the message wrapper, header, etc could be rendered in XML or perhaps as name-value pair extensions to MIME. Similarly the Service Interface could be rendered as Corba, Java, Com, etc.

Note that the definition of the XML or other documents that are transported using the Messaging System as specifically out of scope.

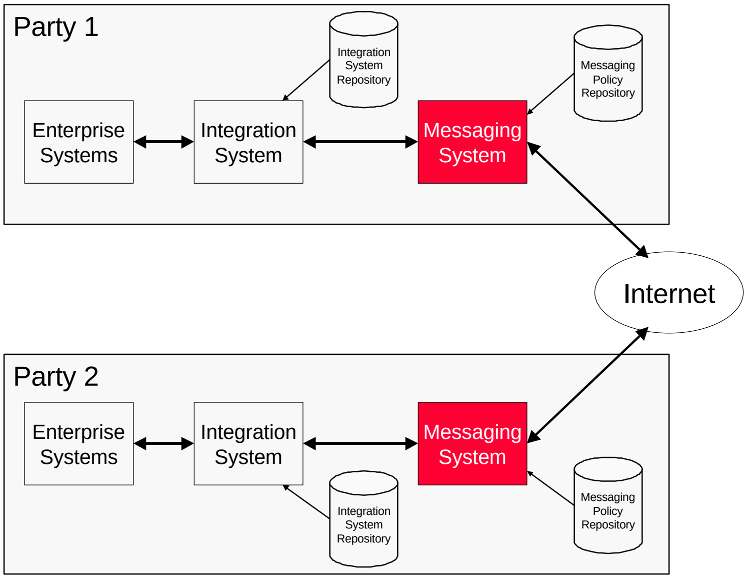


Figure 3 Typical use of Messaging System

Enterprise Systems are *applications* such as accounting systems, ERP systems or other systems that contain data that needs to be communicated with other parties over the Internet.

Messaging Systems are *applications* that manage the exchange of *messages* between the two *parties*. It is agnostic as far as the content or payload within the message is concerned.

Messaging Systems use a **Messaging Policy Repository** to control the behavior of the Messaging System. This contains parameter and other information about how to send *messages* to the other *parties* that the need to be sent messages.

Integration Systems are *applications* that communicate with the Enterprise system and the Messaging System and effectively enables the Enterprise System to exchange data over the Internet. Integration Systems will be required in the short term to integrate existing Enterprise Systems to the Messaging System. Over time, it is probable that Enterprise Systems will be developed or enhanced that can talk natively to the Messaging and other systems such as the system that provides access to data held in the Repository.

Integration Systems use **Integration System Repositories** that contain information on how to format documents and generally communicate between the Messaging System and the Enterprise System

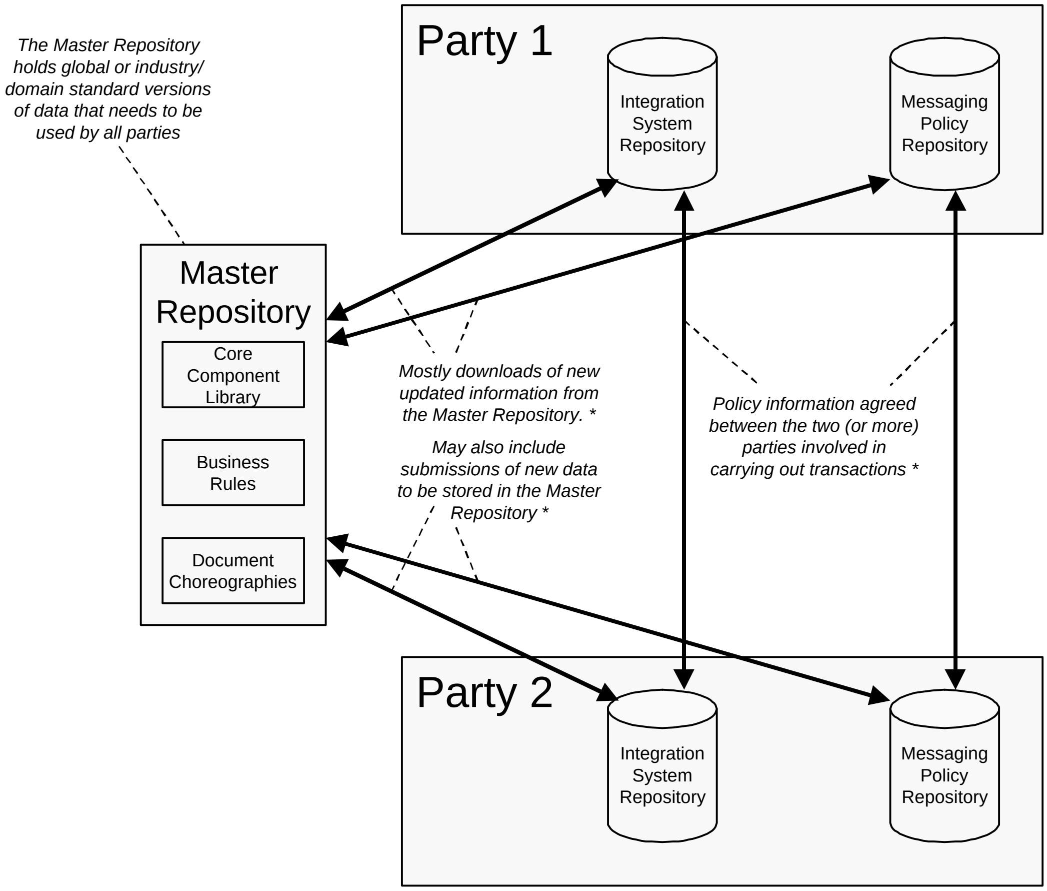

The Master Repository holds global or industry/ domain standard versions of data that needs to be used by all parties
Party 1
Integration System Repository
Messaging Policy Repository
Master Repository
Core Component Library
Business Rules
Document Choreographies
Mostly downloads of new updated information from the Master Repository. *
May also include submissions of new data to be stored in the Master Repository *
Policy information agreed between the two (or more) parties involved in carrying out transactions *
Party 2
Integration System Repository
Messaging Policy Repository

Figure 4 Repositories - Logical Flows of Information

The diagram above illustrates the types of data flow required in order to keep the various repositories in step.

Each of the items marked with an asterisk in the diagram above need Business Processes defined that enable the data in the various repositories to be kept in step.

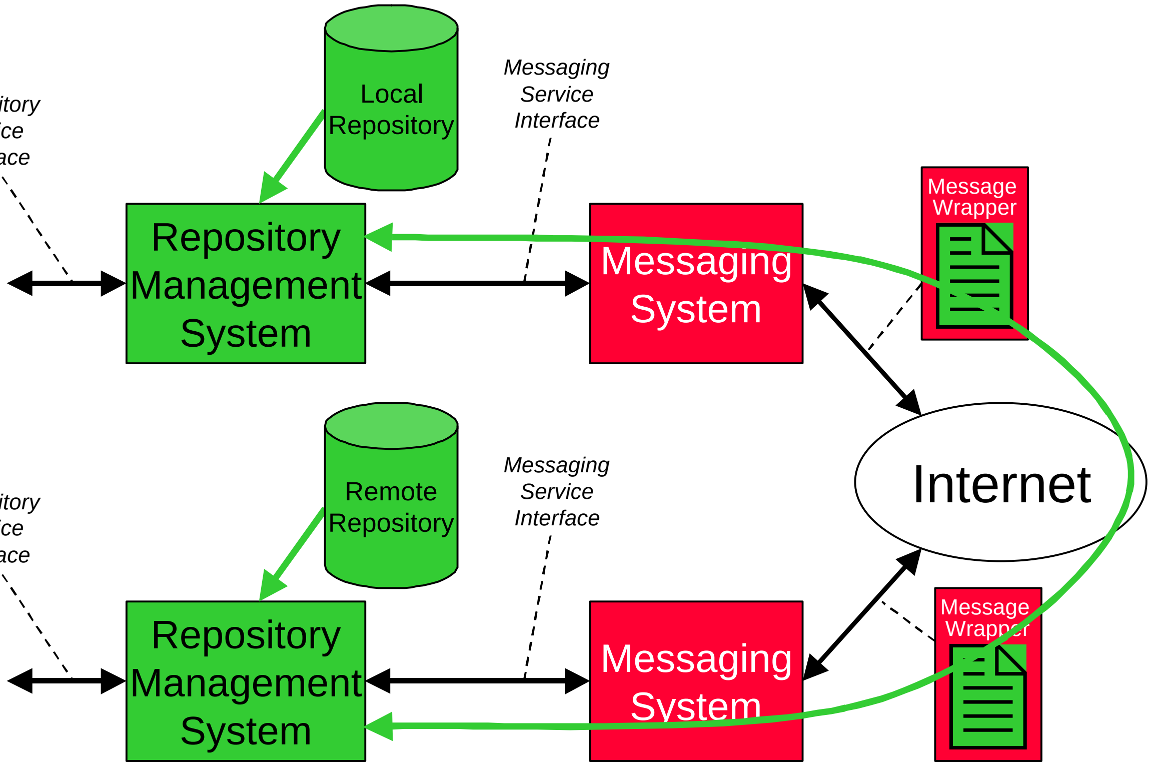

Local Repository
Messaging Service Interface
Repository Service Interface
Repository Management System
Messaging System
Message Wrapper
Internet
Remote Repository
Messaging Service Interface
Repository Service Interface
Repository Management System
Messaging System
Message Wrapper

Figure 5 Repository - Physical Flows of Information

The Repository Management System is no more than just another “System that needs to send/receive data over the Internet” and uses the Messaging System to send receive *messages* in the same way.

The "Messaging System" provides a Service that reliably exchanges messages over the Internet between any two parties. It is completely independent of the content or the payload of the message

The "Repository Management System" provides a Service that can be used to read/update the content of the repository. It would:

- have a Service Interface so that the content of the data in a repository held locally could be maintained, and
- use the Service Interface of the Messaging Service to access the content of repositories held remotely.

Note that the Repository Management System for the Master Repository would also work in the same way.

The Repository Systems used to maintain each of the repositories (Integration, Messaging and Master) may be different from each other and be provided by different vendors. However they should all use the same basic set of documents and document choreography to enable interoperable communication.

The rationale is that the Repository Management Service is really nothing more than a distributed system that reads or updates data on local or remote repositories that has a Service Interface to manage/access the repository content.

This type of "layering" can be used to describe a number of other Service Interfaces that will use the basic messaging service interface, for example:

- a Publish & Subscribe Interface
- a Large Document Transfer Service, for example, to transport multi-MB files reliably by splitting them into several smaller parts that are each transported separately

4 Requirements

This section describes the requirements that the working group aims to meet. They are divided into the following sections:

- Envelope and headers for business documents
- Reliable Messaging and Error Handling
- Messaging Routing
- Security Requirements
- Audit Trails
- Quality of Service
- Platform Independent Interoperability
- Restart and Recovery

4.1 Envelope and headers for business documents

1) *Documents*, expressed either in XML or other electronic formats, shall be able to be wrapped inside a *message envelope* for transporting between the *parties* involved in eCommerce.

2) Multiple *documents*, whether related or not, may be transportable within a single *message envelope*

3) Both the sending and receiving *parties* on a *message header* shall be expressible as:

    a) a physical address (e.g. a URL or an email address) or a logical address (e.g. a DUNS number or EAN) and,

    b) optionally, an address in a human-readable form

4) *Messages* may be transported over many network protocols (e.g. HTTP, SMTP, CORBA, JMQ, MQSeries, MSMQ, etc)

5) *Messages* containing *documents* shall be capable of being globally uniquely identified

6) A *Message* shall identify the *Message* for which it is a response (if one exists).

7) *Message headers* shall contain a timestamp that indicates when the *Message Header* was created

8) *Message headers* may contain a 'maximum lifetime' indicator that specifies that maximum amount of time that a *message* should be considered 'alive' after it is sent.

9) *Message headers* may contain an address to which response messages can be routed; actual use of this property by sending and receiving *services* is optional

10) *Message headers* may contain an indication of the priority of a *message*

11) *Message headers* may contain the name of an administrative address to which acknowledgement messages can be routed.

12) *Message headers* should allow application specific routing headers in the *message.*

13) A *Message Manifest* shall detail all parts contained in the envelope and references to external document sources if required.

4.2 Reliable Messaging and Error Handling

1) *Messages* shall be capable of being delivered from a sending *party's Service* to a receiving *party's* Service so that:
    a) delivery occurs at most once[1].
    b) failure to deliver shall be reported, if the sending party requires it.
    c) inability to send a *document* may be notified to the *party* that sent the document
    d) if an *application*/business level *response message* is not received within expected timescales then there shall be mechanisms that support recovery
    e) the correct sequence in which related messages are sent can be identified
    f) recovery from failure to receive a *response message* should include:
        i) how the "expected timescales" after which recovery starts are specified
        ii) descriptions of the *messages* sent to carry out the recovery

2) *Error messages* should be capable of reporting on:
    a) errors associated with the underlying transport protocol, e.g. HTTP
    b) errors in the *message wrapper, message header* or *message routing information*
    c) errors with the way *documents* are wrapped inside their *message envelopes*
    d) errors associated with failed attempts at reliable once-only delivery of *messages*
    e) errors in the *documents* that are being transported
    f) errors in the sequence in which *messages* are exchanged
    g) abnormal errors with the *services* that processed the *documents* (e.g. the service crashed) and
    h) business failures where the *service* completed but did not realize its hoped for outcome (e.g. out-of-stock)

3) Inquiries should be possible to determine why *Message Sets* failed, (see Message Set Status Inquiry below).

[1] There are four types of delivery:

- at most once - the message is delivered either zero or one times from *service* to *service*
- exactly once - message is delivered once and only once from *service* to *service*
- at least once - message is delivered one or more times from *service* to *service*
- unknown - message is delivered zero more times from *service* to *service*

At most once means that the receiving party's messaging system shall ensure that multiple copies of the same message received, results in a single delivery of the message to the receivng party's service.

4.3 Message Routing

1) *Messages* may be sent using a variety of methods:
    a) to a single *party*, e.g. by specifying a URL
    b) to multiple *parties*, either by:
        i) specifying a list of URIs in the *Message Header*, or
        ii) a distribution list held separately from the header
    c) to an agent or intermediary for forwarding to the next *party*
2) Individual *messages* shall be capable of routing serially or in parallel with other related *messages*
3) Publish and Subscribe
    a) *Messages* may be distributed to the members of a list of *parties* using a "Publish and Subscribe" mechanism
    b) the anonymity of the subscriber may optionally be maintained

4.4 Security Requirements

1) For non-repudiation, message integrity and authentication purposes, the following are requirements:
    a) *Documents* and/or *message headers* may be digitally signed
    b) The signature over the *documents* or *message headers* shall be independent of the transport protocol used[2]
    c) A single digital signature may be used to bind together *documents* either:
        i) within the same *message*
        ii) in another *message*[3]
        iii) somewhere else (for example the content at a URL)[4]
    d) Signatures on digitally signed *documents* may be used to:
        i) verify the authenticity of the *party* that is the sender,
        ii) provide non-repudiation of origin or receipt, and
        iii) ensure that the content of the message has not changed
2) For privacy and confidentiality purposes:

[2] The rational behind this is that:

- we need to be able to support multiple transport protocols and therefore reliance on transport level protocols would mean that transport specific signature handling would be required
- we need to be able to persist the signature for later checking or re-use, after the message has been received

[3] The can be used, for example, to bind one message to an earlier message and therefore provide an audit trail

[4] An example of where this might be used is to bind together an Invoice send in a message with the terms and conditions held somewhere in an HTML file on the web

   a) All or part of the *documents* in a *message* may be encrypted prior to sending

   b) *messages* may be encrypted during transportation using a transport protocol

3) Secure timestamps:

   a) *documents* or *messages* may be time stamped securely with a digital signature

   b) secure time stamps may be generated by a trusted third party

   c) timestamps shall be recorded in a location independent way (e.g. UTC).

4.5 Audit Trails

1) The set of ~~related~~ *documents* and *messages* that are contained within a *Message Set,* shall be::

   a) globally uniquely identified,

   b) related to one another.

2) Two or more *Message Sets* that are related to one another should be capable of being linked together by enabling one *Message Set* to refer to another *Message Set's* Message Set identifier.

3) A trace or path through the *services* and *parties* through which *documents* have passed should be identifiable and analyzable after the event

4) Digital signatures may be used to bind the *documents* and *Message Sets* in the sequence in which they were used.

4.6 Quality of service

The Quality of Service of the interaction between two Services is defined in a *Transport Service Level Agreement (TSLA)*. The parameters in a TSLA vary depending on the nature of the Service.

Parameters must be present in every TSLA that: .

1) support *Session* based and Long Term Transactions

2) enable recovery from failure to receive an anticipated response(s) to a message

3) enable a Receiving Service to inform a sender of a message of the Receiving Service's expected maximum *Response Time(s)*

4) enable a sender of a message to inform the recipient of a message, of the Response Time(s) that the sender expects

5) enable a sender of a message to discover if a Receiving Service is operational and therfore able to receive messages

6) enable a sender of a message to discover the hours of operation of a Receiving Service The hours of operation is the period of time that the service is available to process the message

7) enable a Receiving Service to indicate to the sender of a message that it is too busy to process a message within expected timeframes. This supports congestion management

8) enable a sender of a message to discover from a Receiving Service the current status of a Message Set.[5] This is *Message Set* Status Inquiry.

[5] This is particularly relevant if Asynchronous processing is being used

9) enable the Sending and Receiving Parties to discover and agree:
    a) the document choreographies that can support their processing requirements
    b) the parameters that control how the parties will use cryptography
    c) how they will achieve reliable messaging and error handling when required
    d) the transport protocols to be used
10) TSLAs may be negotiated between two Parties that apply to:
    a) an individual *message*
    b) an individual *message set*
    c) all messages associated with one or more services
    d) all interactions between two parties

4.7 Platform Independent Interoperability

1) Servers/systems that support the exchange of documents shall be treated as "black boxes"[6]
2) The method used to transport documents shall be completely independent of:
    a) the hardware used by the server/services at each end
    b) the software or systems architecture of the server/services at each
    c) the language used for implementation of systems and *application*s.
3) Support for a *service* shall be expressible solely in terms of the type and sequence in which *documents* (and their *message envelopes*) are to be exchanged
4) The ebXML Transport, Routing and Packaging specifications shall be suitable for implementation on hardware that varies from a very simple device to a large multi-processor/system complex

4.8 Restart and recovery

1) If a *service* that accepts *messages* becomes temporarily unavailable after starting a *Message Set* it shall be possible to recover from the failure and deliver the message once the *service* is available
2) If a *service* that accepts *messages* is temporarily unavailable before starting a *Message Set* then it shall be possible to recover from the failure and deliver the message once the *service* is available
3) If the delivery of a *message* is considered not possible by the originally intended method, then
    a) alternative methods of delivering the *message* may be used[7] if available, and/or
    b) the end state of the Message Set shall be capable of rollback to a consistent state.

[6] This means that the sender and recipient of messages shall agree beforehand the document and message structures that will be used

[7] An example would be delivery by SMTP or CORBA if HTTP was not possible

4.9 Protocol Extensibility

1) The protocol shall be extensible to support (by use of protocol versioning):

    c) additional types of data in message headers and message routing information

    d) new values for codes[8]

    e) new ways and methods of exchanging data

5 Definitions

The following are a list of definitions of the terms associated with the transport of messages over the Internet. They are derived initially from work being done within the IETF.

It is split into two sections:

- Documents, Parties, Messages and Document Exchanges, and
- Services and Message Sets

Words or phrases that are defined elsewhere are highlighted in *italics*.

5.1 Documents, Parties, Messages and Document Exchanges

5.1.1 Overview

This section describes how *Parties*, such as buyers and suppliers, customers and merchants, can transmit *Documents* contained in *Messages* in order to request execution of *Services*.

All the *Documents* and other data in a *Message* are contained within an outermost *Message Envelope*.

A *Message* can optionally include *Digital Signatures* so that:

1) the identity of the *Party* sending the *Message* can be authenticated

2) any changes to the *message* and the *documents* they contain can be detected.

Services are requested by sending one or more *Documents* in a *Request Message* to a *Party* who then:

1) processes the *Request Message* by carrying out a *Service* and

2) optionally generates a *Response Message* indicateing result.

At a minimum a *Document Exchange* consists of a *Request Message* and an optional *Response Message* although there might be additional *Exchange Messages* between the *Request Message* and the *Response Message*.

Error Messages are used to report permanent or transient problems or errors in a *Message*.

More detail is provided below.

5.1.2 A Document

A *Document* is any data that can be represented in a digital form.

[8] It is likely that XML Schema from the W3C will be able to provide extensibility for new types of data and values for codes.

Examples of *Documents* include:

1) a set of XML Elements
2) an XML Document
3) an HTML Document
4) a word processing file
5) an Adobe Acrobat PDF file
6) a binary file
7) part of larger document.

5.1.3 Party

A *Party* is a company, organization or individual or other entity that can generate, receive or relay *Documents*.

Examples of a *Party* include:

1) a Merchant
2) a Customer
3) a Lawyer
4) a Bank
5) a government department or agency
6) an intermediary or agent
7) a software agent

A *Party* is also used to refer to systems or servers that are carrying out *Services* or processes on behalf of a *Party*.

5.1.4 Message

A *Message* is data that is sent from one *Party* to another. A *Message* consists of information such as:

1) a *Message Header* that indicates who sent, who should receive and the context for sending the message
2) *Message Routing Information*, that indicates how the message should be / was delivered
3) *Digital Signatures* to:
    a) bind the data in the message, or elsewhere, together, and
    b) ensure that changes to the data can be detected
    c) enable authentication of the sender of the message
4) *Documents* which are the business data that actually needs to be sent

All the data in a *Message* is contained within a *Message Envelope*.

Examples of a *Message* include:

1) a Purchase Order that is sent by a buyer to a supplier
2) an Invoice that is sent by the supplier back to the buyer

3) a request to make a payment of $50 sent to a Credit Card acquirer

4) the authorization received from a Credit Card acquirer as a result of making a payment

5) Status Data indicating the success or failure of a Service

5.1.5 Message Header

A *Message Header* is an XML construct that contains the additional data that needs to be associated with the *Documents* in a *message* so that they can be sent to and successfully processed by a *Party*. It can contain information such as:

1) Message Set Identity data to identify the set of *Messages* that are related to one another through one or more *Document Exchanges*

2) Message Identity data to enable the *Message* to be identified and referenced within the *Message Set*

3) a *Message Manifest* to identify the documents, other than the *Message Header*, that are contained within the same *Message Envelope*

4) Action Data to indicate the *Service* that is being sent the message and the reason for sending

5) Organization Data that describes one or more of:

    a) the Sender organization that sent the *Message*

    b) the Recipient organization(s) that ought to receive the *Message*

    c) the Authorizing organization(s) that provide evidence that a requested *Service* should be carried out.

6) Status Data that describes the results of carrying out a *Service*.

5.1.6 Message Manifest

The Message Manifest contains references to the other documents, apart from the Message Routing Information document, that are contained within the same Message Envelope.

The purpose of the Message Manifest is to facilitate locating and validating that all required Documents contained within the Message Envelope are present.

Examples of the types of documents that might be referenced by a Message Manifest include:

1) a Purchase Order

2) a Purchase Order and a picture of the requested goods

3) a Purchase Order and a digital signature

5.1.7 Message Routing Information

Message Routing Information contains data that indicates the path that should be or was taken by a *Message* in reaching its ultimate destination.

5.1.8 Digital Signature

A *Digital Signature* is a cryptographic signature over[9] data contained in a *Message*, or elsewhere that are addressable via URIs, that permits the authenticity of the signer of the data to be determined, and helps detect if the data in the *Message* has changed.

5.1.9 Message Envelope

A *Message Envelope* is the outermost container for a Message. It can be such things as:

1) an XML Document, or

2) a multi-part MIME message

5.1.10 Request Message

A *Request Message* is a *Message* sent from one *Party* to another *Party's Service* with the intent that the other *Party* act upon the data in the *Request Message* by carrying out the *Service*.

5.1.11 Acknowledgement Message

An Acknowledgement Message may sent as a response to any *Message* (apart from an Acknowledgement Message) to indicate that the Message has been received[10].

5.1.12 Checked OK Message

A Checked OK Message may be sent in response to a Request Message to indicate that the content of the message has been validated and no errors were found

5.1.13 Response Message

A *Response Message* is a *Message* that is generated by the *Service* that received a *Request Message*. It is produced as a result of carrying out the requested *Service*. It is the last *Message* in a *Document Exchange* unless the *Message* contains errors.

Response Messages are sent back to the sender of the *Request Message*.

5.1.14 Document Exchange

A *Document Exchange* is a generic term for either a *Simple Document Exchange* or a *Multiple Round Trip Document Exchange*. Examples of Document Exchanges are contained in section 6 Examples of Document Exchanges

[9] A digital signature represents a string of binary digits of arbitrary length created by using a cryptographic key known only to the party sending a message. The string is composed of an encrypted digest of some or all of the data in the message or in another location addressable by a URI. It is accompanied by some method (such as a digital certificate)of identifying to the party receiving the message, what key can be used to validate the digest against the original data.

[10] It is recommended that messages are saved in some type of persistent storage before they are acknowledged.

5.1.15 Simple Document Exchange

A Simple Document Exchange consists of:

1) a *Request Message* sent from one *Party* to a second *Party*, followed by
2) an optional *Acknowledgement Message* sent by the second party back to the first party, followed by
3) an optional *Checked OK Message* sent by the second party back to the first party followed by
4) an optional *Response Message* that is returned as a result of processing the Request Message.

Examples of instances of a *Simple Document Exchange* include:

1) a Purchase Order sent by a buyer to a seller and the acknowledgement from the seller of its receipt
2) a Purchase Order sent by a buyer to a seller and the Invoice that is sent back as a result of fulfilling the order
3) sending a document for review by a lawyer followed by the legal opinion that is sent back as a result

5.1.16 Multiple Round Trip Document Exchange

A *Multiple Round Trip Document Exchange* consists of:

1) a *Request Message* sent from one *Party* to a second *Party*, followed by
2) a series of *Exchange Messages* that are exchanged between the two *Parties* until finally
3) the second *Party* generates and sends a *Response Message* back to the first *Party*.

Examples of *Multiple Round Trip Document Exchanges* include:

4) the exchange of messages required to make a payment using payment method protocols such as [SET] or [Mondex]
5) the exchange of messages required to negotiate an agreement on terms and conditions.

5.1.17 Exchange Message

An *Exchange Message* is a *Message* that is sent between one *Party* and another after the sending of the initial *Request Message* and before the sending of the final *Response Message*.

Examples of *Exchange Messages* include:

1) intermediate messages that are part of a Payment Protocol
2) a counter offer to an offer made as part of a negotiation.

5.1.18 Error Message

An *Error Message* is a *Message* that reports on a problem in an earlier *Message* that prevents the earlier *Message* from being processed in a normal way.

Examples of an *Error Message* include:

1) an *Error Message* reporting that an XML document was invalid or did not conform to its XML schema

2) an *Error Message* reporting a Transient Error that the Server processing a *Message* is busy and therefore the original *Message* should be resent at a later point in time

3) an *Error Message* that reports on an error in the underlying transport protocol.

5.2 Services and Message Sets

5.2.1 Overview

A *Service Definition* describes a process that can be carried out by a *Party.* It consists of either a *Document Exchange* or a set of *Sub-Services*. Each *Sub-Service* is a *Service* in its own right. So, at the lowest level, all *Service Definitions* are described in terms of a *Document Exchange*.

The dependencies between the *Sub-Services* in a *Service* is described in a *Sub-Service Choreography*.

An instance of the execution of a *Service Definition* is called a *Message Set*.

The parameters that define how the transport of messages is managed and controlled is specified in a Transport Service Level Agreement (TSLA)

More detail is provided below.

5.2.2 Service Definition

A *Service Definition* describes a process that can be carried out by a *Party* as a result of receiving a *Request Message* that requests the execution of that *Service*.

A *Service Definition* can consist of either:

1) a Document Exchange, or

2) a set of Sub-Services

Examples of *Service Definitions* include descriptions of:

1) a Purchasing Service that enables a customer to purchase goods on-line

2) an Order Processing Service that processes an Order and generates a response as a result

3) a Payment Service that accepts a payment and provides a receipt

4) a Fulfillment Service that fulfills an order at the request of a Merchant.

5.2.3 Sub-Service

A *Sub-Service* is a *Service* that is executed at the request of and as part of another *Service*.

Examples of *Sub-Services* include:

1) a payment service that occurs as part of a purchase

2) a tax calculation service that calculates the tax due as part of an order processing service.

An example of how services, sub-services and document exchanges relate to one another is illustrated by the diagram below.

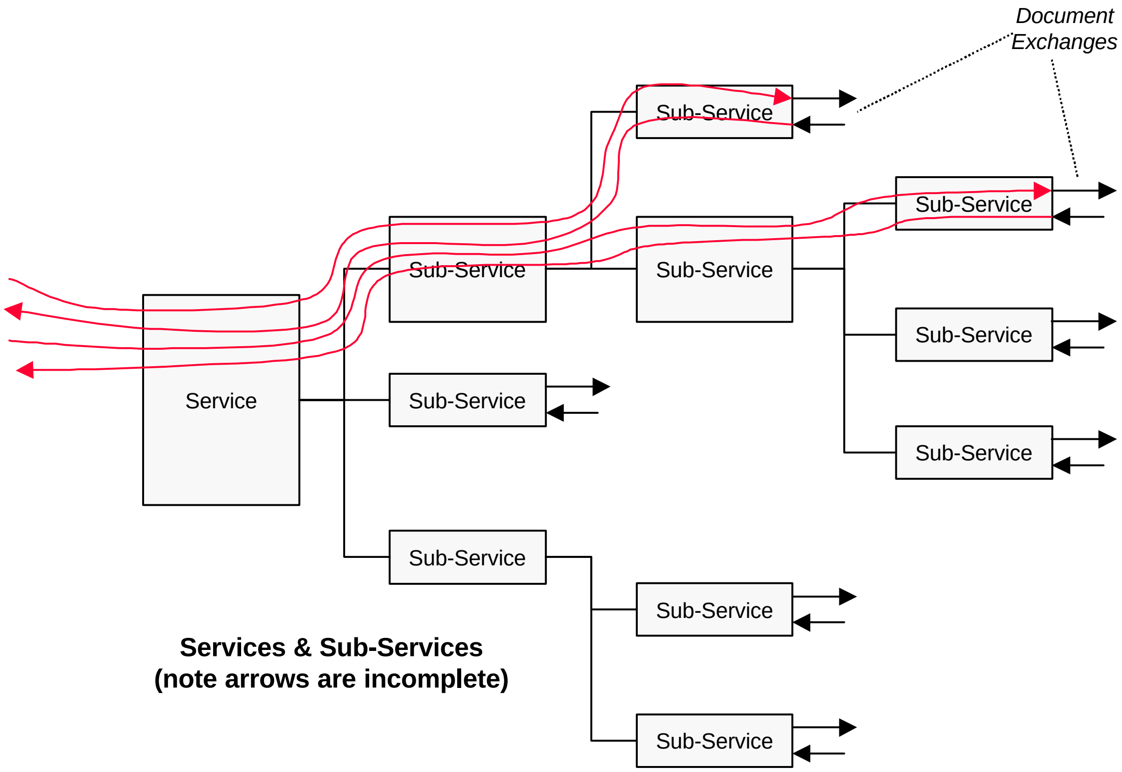


Figure 6 Services and Sub Services

5.2.4 Sub-Service Choreography

A *Sub-Service Choreography* is a description of the dependencies that control the sequence and choices that determine which *Sub-Services* are executed when carrying out a *Message Set*.

The *Sub-Services* in a *Service* will have dependencies between them. Dependencies can be:

1) Serial. One *Sub-Service* shall start only after the completion of another *Sub-Service*

2) Alternative. One *Sub-Service* may be executed as an alternative to another

3) Iterative Loop. A *Sub-Service* may be repeated a variable number of times

4) Conditional. The execution of a *Sub-Service* is conditional on the state of another *Service*. This may be used in conjunction with Serial, Alternative and Iterative Loop dependencies.

5) Parallel. A *Sub-Service* may execute in Parallel with another *Service*

6) Concurrent. A *Sub-Service* shall Execute at the same time as another *Sub-Service*.

An example of a simple *Sub-Service Choreography* is a Purchase Service that consists of three *Sub-Services*:

1) an Offer Service that conveys an Offer for sale of goods. This *Sub-Service* has no dependencies and therefore starts first

2) a Payment Service that carries out the Payment which has a Serial dependency on the Offer Service

3) a Delivery Service that delivers the Digital Goods, that has a Serial Dependency on the Payment Service

5.2.5 Application

An Application is software that may implement a *service* by processing one or more of the *messages* in the *document exchanges* associated with the *service.*

5.2.6 Message Set

A *Message Set* is an instance of the execution of a *Service*[11].

Examples of a *Message Set* include:

1) a Purchase Message Set that buys a Company Report for $20. It consists of three Sub-Service instances:
    a) an Offer Service instance to buy the Company Report for $20
    b) a Payment Service instance that accepts a Payment for $20 using a credit card, and finally
    c) a Delivery Service instance that delivers the Company Report as an HTML web page.
2) a Buying Service that consists of the following Sub-Services:
    a) three Price Negotiation Service instances that negotiate the price of a Photocopier
    b) a Purchase Order Service instance that places the order for the Photocopier.

5.2.7 Transport Service Level Agreement

Transport Service Level Agreement (TSLA) consists of information mutually agreed between two paties that manages and controls the ebXML "transport" software that sends and receives messages.

Transport software is software that is constructed according to the specifications produced by the ebXML Transport, Routing and Packaging Project Team.

Examples of information in a TSLA include:

1) timeout parameters
2) retry counts

[11] There are several different meaning that have been associated with Message Sets:

- "ACID" Message Sets (TBD) A Message Set can be considered a collection of actions with the following properties:
    - **Atomicity**. A Message Set's changes to the state are atomic: either all actions happen or none happen.
    - **Consistency**. A Message Set is a correct transformation of the state. The actions taken as a whole do not violate any of the integrity constraints associated with the state. This requires that the Message Set be a correct program.
    - **Isolation**. Even though Message Sets execute concurrently, it appears to each Message Set T, that others executed either before or after T, but not both. In other words, each Message Set is isolated from any others.
    - **Durability** Once a Message Set completes successfully (commits), its changes to the state survive failures.
- "EDI" Message Sets - "The information included in a Message Set set is, for the most part, the same as the information in a conventionally printed document. A Message Set set is the data that is exchanged in order to convey meaning between parties engaged in EDI "Conversational" Message Sets - A conversation is a sequence of related Message Sets between two parties separated in time. A complete "unit of business" for example, the negotiation of a purchase, placement, confirmation, payment and delivery of goods, may be represented as multiple Message Sets in a longer running conversation." From DISA publication titled "Introduction to EDI", (ASC X12S/94-190)
- "Read-only" Message Sets - a Message Set that consists of a document exchange where the information is obtained from a service without changing the state of the service

3) security parameters

4) respponse addresses

5.3 Miscellaneous

1) A *Session* based *Message Set* is where a *Document* is sent to a *Party* which results in an immediate response of another *Document*. These are synchronous in nature.

2) A long term *Message Set* is where a *Document* is sent to a *party* and, possibly, a simple acknowledgement is sent back immediately. The *Document* that is the "business" response to the original *Document* is then sent some time later

3) *Response Time* is the time taken by a *Service* to process a *Message* and generate a response[12].

6 Examples of Document Exchanges

The following diagrams provide an non-exhaustive list of the different types of template sequences in which messages can be exchanged.

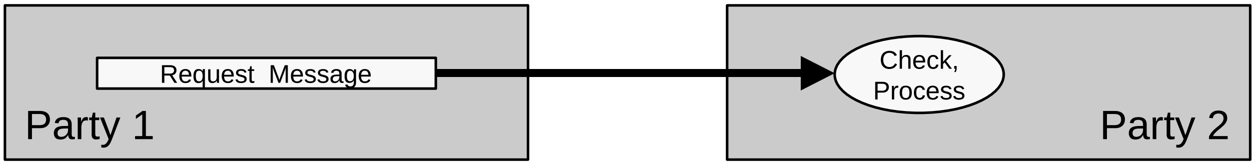


Figure 7 Simple Request

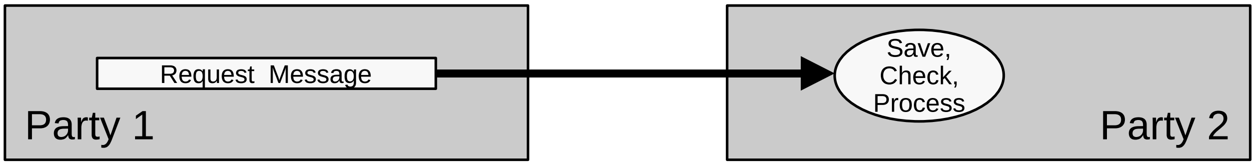
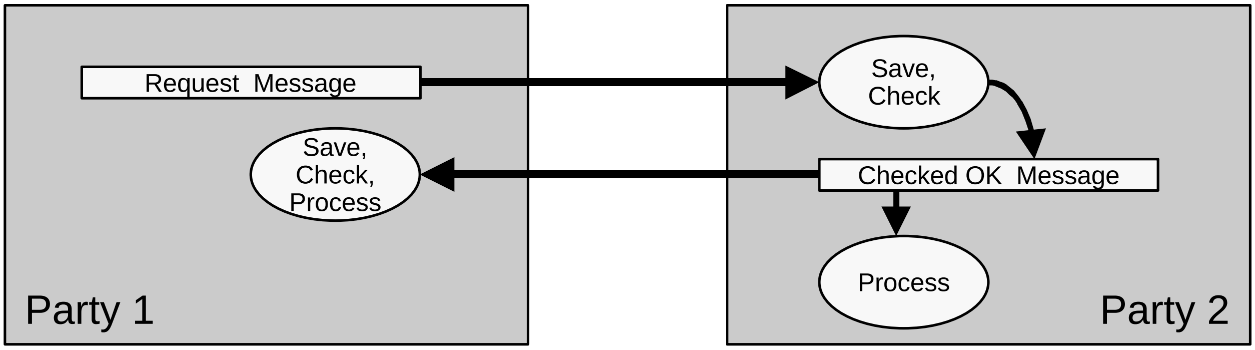


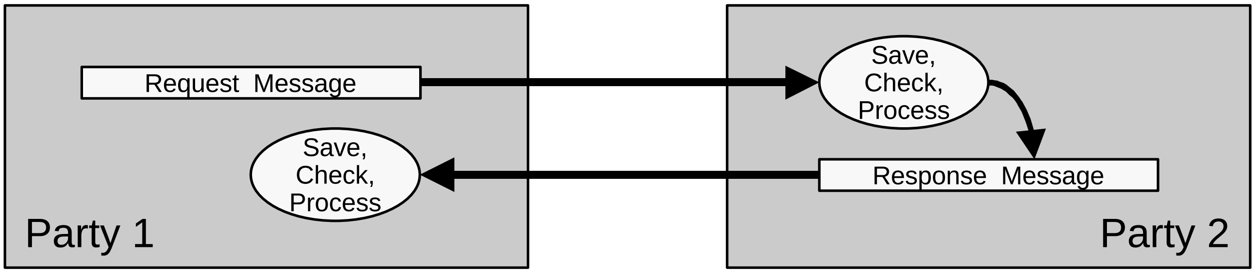
Figure 8 Simple Request with Save

[12] The Response Time perceived by the sender of a message will be different from the response time perceived by the recipient of the message to process it since the first includes the transmission time of the message (and it's response) whereas the second does not.



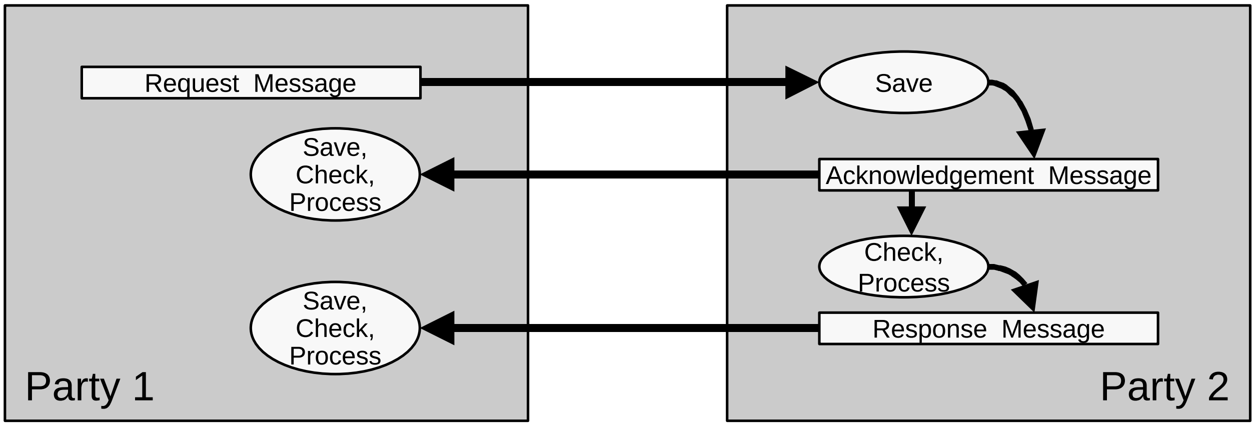
513

514 **Figure 9 Simple Request and Checked OK, No Response required**



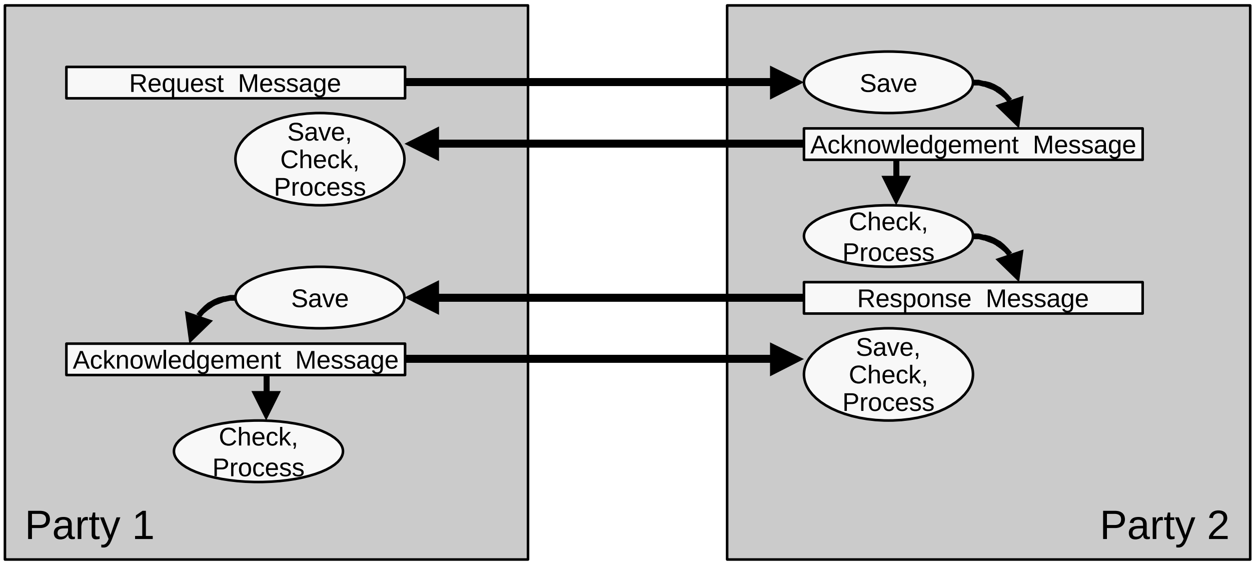
515

516 **Figure 10 Simple Request Response**

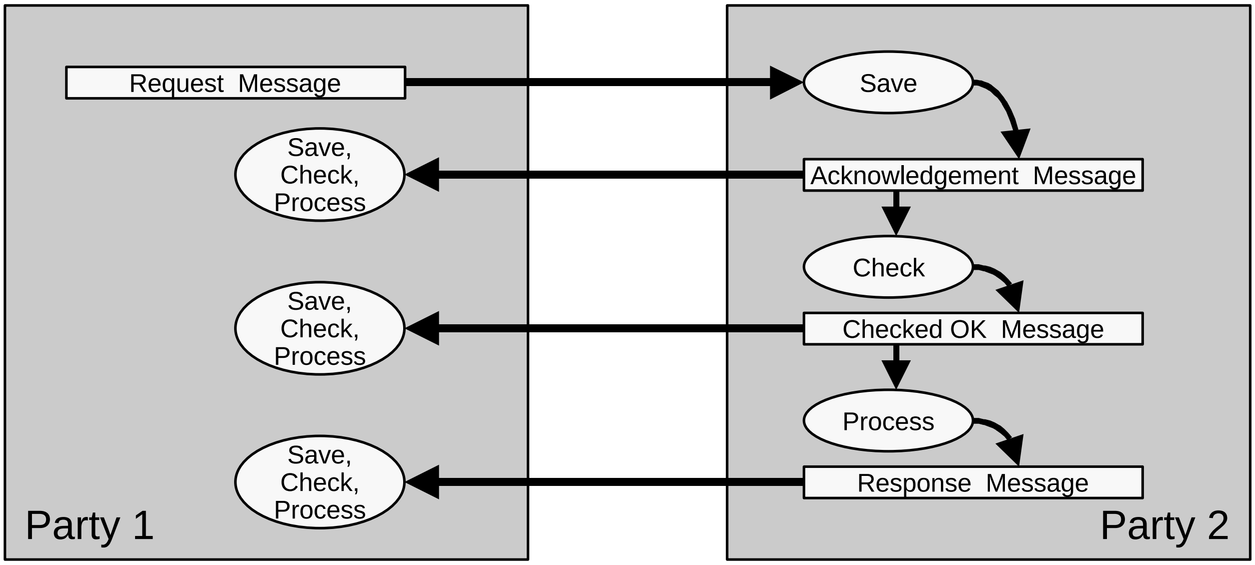


517

518 **Figure 11 Simple Request with Acknowledgement and Response**

519 

520 **Figure 12 Simple Request Response - both with Acknowledgement**

521 

522 **Figure 13 Request with Acknowledgement, Checked OK and Response**

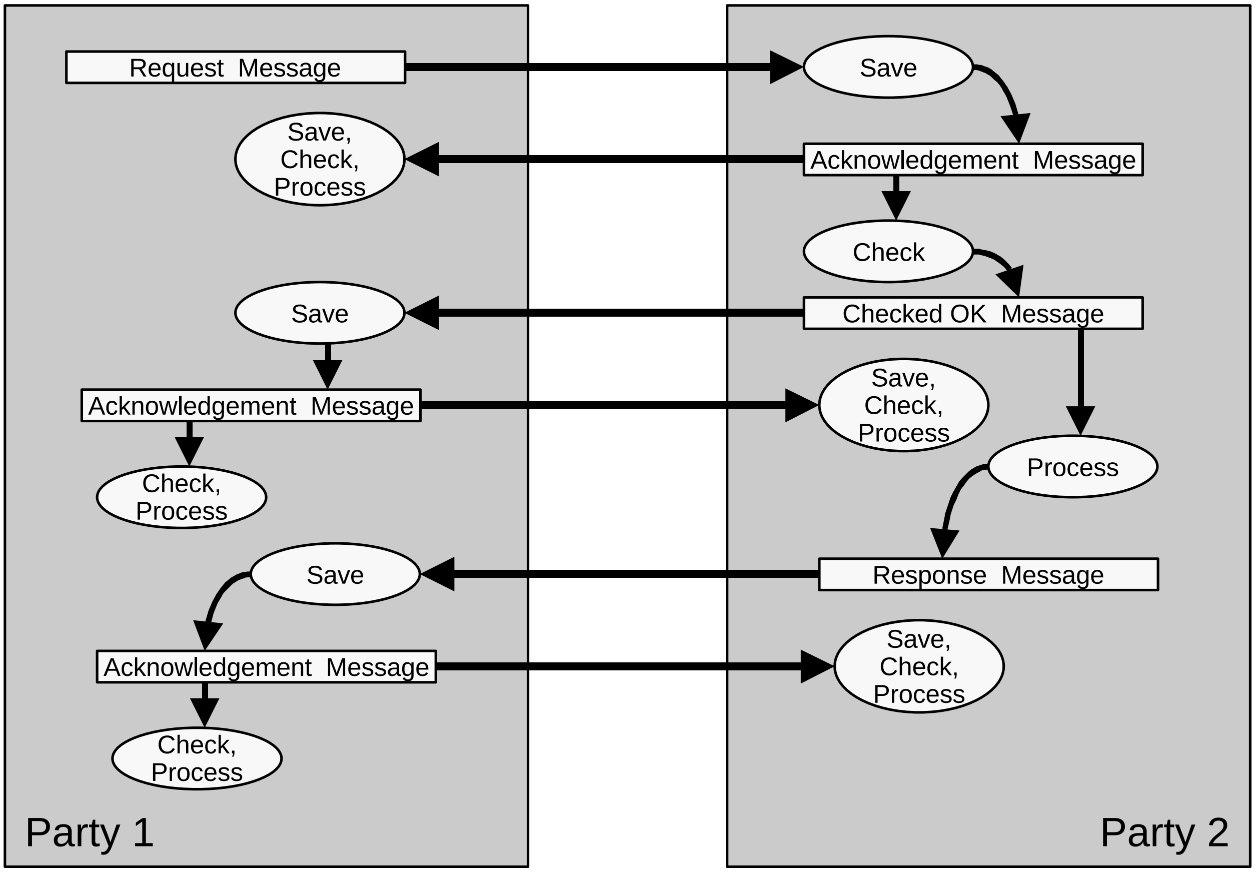


Figure 14 Acknowledgements with Everything

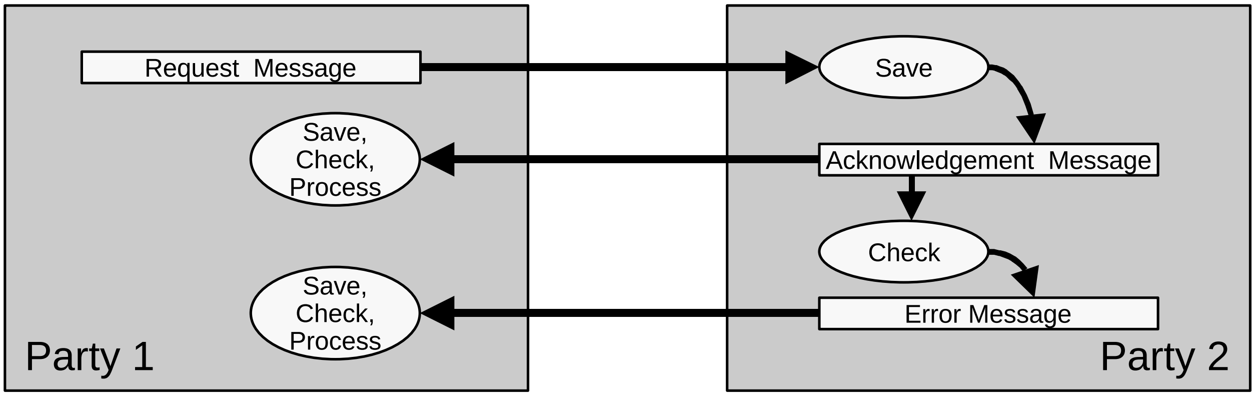


Figure 15 Request Message with Error

7 References

No references.

8 Acknowledgements

This document is a collective development effort of all the members of the Transport, Routing and Packaging project team.