Title: *Electronic data interchanges on the Internal Electricity Market*
Ref.: *AF_284*
Version: *0.4*
Date: *28/04/2014*

DOCUMENT CHANGES

Version	Change description
0.0 (28/08/2012)	Initial version
0.1 (06/02/2013)	Base messaging on IEC 61968-100
0.2 (19/04/2013)	Add QueryData service
0.3 (19/12/2013)	Clarify some terms and definitions. Add ServerTimestamp to List Remove CanonicalIdentification. Add binary file name.
0.4 (28/04/2014)	Remove MTOM to better align with IEC 61968-100 Improve and simplify Digital Signature Add Java code examples Add generic QueryData parameters

INDEX

1 Scope

1.1 General

This specification defines the services needed to support the electronic data interchanges between different actors on the European Energy Market for Electricity (EME) in a fast (near-realtime), and secure way.

Web Services (in WSDL) will be specified for the defined services, applying the Basic Webservice Pattern implementation profile from IEC 61968-100.

The content of this document has been submitted to IEC as proposal 57/1376/NP, and is in the process of becoming a Technical Specification (IEC 62325-504 TS Ed.1).

The publication of IEC 62325-504 as a Technical Specification will supersede this document.

1.2 Overview

The services needed to support the electronic data interchange on the European Energy Market for Electricity are:

- List Messages. This service is used by a client application identified with the credentials of an EME Actor to request a list of messages available on the server for retrieval.
- Get Message. This service is used by a client application identified with the credentials of an EME Actor to request a specific message available on the server.
- Put Message. This service is used by a client application to send a message, usually providing data related to a Market Participant in the Energy Market for Electricity, to the server for processing.

.

2 Normative References

The following referenced documents are indispensable for the application of this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

WS-I Basic Profile 1.1, http://www.ws-i.org/Profiles/BasicProfile-1.1-2006-04-10.html

WSDL, *Web Services Description Language (WSDL) 1.1*

IEC-40210, *W3C SOAP Version 1.2 Part I: Messaging Framework (2nd Edition)*

XML Schema 1.0: XML Schema Language Part 1: Structure, W3C Recommendation 28 October 2004; XML Schema Language Part 2: Data Types, W3C Recommendation 28 October 2004

XML Signature Syntax and Processing (Second Edition) http://www.w3.org/TR/xmldsig-core

IEC-62325 Part 451-1: *Framework for energy market communications - Acknowledgement business process and contextual model for CIM European market.*

IEC-61968-100: *Implementation Profiles for IEC 61968*

3 Terms, definitions and namespaces

3.1 Terms and definitions

3.1.1 Message Identification

Alphanumeric string that represents the name of a message in the system.

3.1.2 Version

Number that represents the message version. The range of values is from 1 to 999.

3.1.3 Application Time Interval

Time interval when the message payload applies.

3.1.4 Server Timestamp

Date when the message is received by the server (input messages) or when is made available by the server (output messages)

3.1.5 Message Type

Type of the message payload as defined in IEC 62325 Part 451-n (Schedule, NTC, Acknowledgement, etc.)

3.1.6 Message Code

This number identifies a message in the server in a uniquely way. For a given message codes "x" and "y", if "y" > "x" then "y" is a newer message. If "y" < "x" then "y" is an older message. Finally if "y" = "x", then both messages are the same.

3.1.7 Data Owner

Person or entity that is responsible for the information contained in the message (payload). Usually corresponds with the sender_MarketParticipant.mRID field in the IEC62325-451-n series.

3.1.8 Data Provider

Person or entity that is responsible for establishing a connection with the server and sending the message (payload).

3.1.9 M/O/C

Mandatory / Optional / Choice (choose one). Cn indicates "Choice n" and if several optional attributes have the same number "n" in Cn, it means all of them shall be present if this is the selected choice.

3.1.10 Status

Corresponds with the main reason code of the Acknowledgement message associated with a message as per IEC-623235-451-1. The status of messages without Acknowledgement (publication or incoming message still being processed) will be "OK".

3.2 Namespaces

This document uses these prefixes and namespaces:

a) cmsg (urn:iec62325.504:messages:1:0): The target namespace of the messages defined in this document.

b) wss (urn:iec62325.504:wss:1:0): The WSDL target namespace for this document.

This document refers to these other prefixes and namespaces:

a) wsdl (http://schemas.xmlsoap.org/wsdl): This contains the W3C WSDL 1.1 schema.

b) xs (http://www.w3.org/2001/XMLSchema): This contains the W3C XML Schema definition.

c) soap (http://schemas.xmlsoap.org/wsdl/soap): This contains the W3C SOAP bindings for WSDL 1.1.

d) soap12 (http://schemas.xmlsoap.org/wsdl/soap12): This contains the W3C SOAP bindings for WSDL 1.2

e) ds (http://www.w3.org/2000/09/xmldsig#): This contains the XML Digital Signature Schema definitions

f) xmime (http://www.w3.org/2005/05/xmlmime): This contains the W3C Media Content Description of Binary Data in XML

g) msg (http://www.iec.ch/TC57/2008/schema/message): This contains the IEC 61968-100 schema definitions.

4 Conformance

4.1 General

This Clause specifies the conformance requirements for an application and a server to conform to this specification

4.2 Client Application Conformance

In order to conform to this specification a client application shall:

a) Support the following services:
   - List Messages and all of the mandatory aspects of this service as specified in Clause 5
   - Get Message and all of the mandatory aspects of this service as specified in Clause 5
   - Put Message and all of the mandatory aspects of this service as specified in Clause 5

b) Send and receive XML Instance documents according to the XML Schema specified in Clause 7 in this specification for the services listed in a).

c) Use the WSDL definitions, SOAP bindings, and operations specified in Clauses 8 and 9

d) Be able to access the server via HTTPS, using a client digital certificate recognized by the server for the purposes of establishing the https communication and creating the digital signature as specified in Clause 10

4.3 Server Conformance

In order to conform to this specification a server shall:

a) Support the following services:
   - List Messages and all of the mandatory aspects of this service as specified in Clause 5
   - Get Message service and all of the mandatory aspects of this service as specified in Clause 5
   - Put Message service and all of the mandatory aspects of this service as specified in Clause 5

b) Send and receive XML Instance documents according to the XML Schema specified in Clause 7 in this document for the services listed in a)

c) Use the WSDL definitions, SOAP bindings, and operations specified in Clauses 8 and 9

Provide access to the server via HTTPS, and be able to asses that the client digital certificate is valid and that the digital signature as specified in Clause 10 is correct.

5 Service Definitions

5.1 List Messages

5.1.1 General

The List Messages service is used to obtain a list of available messages for the client according to a given filter (parameters).

The main filter shall be one of the following:

- application date of the returned messages
- Server Timestamp of the returned messages
- internal numerical code of the returned messages

Additional optional filters include:

- Message Identification
- Message Type
- Data Owner

The returned list of messages will comply with the main filter selected and also with all the optional filters requested, and will include the following information related to each message:

- Internal numerical code representing the message in the server
- Message Identification
- Message Version
- Status
- Application TimeInterval
- Server Timestamp
- Message Type
- Data Owner

5.1.2 Service Request

Parameter Name	Type	M/O/C	Description
StartTime	dateTime	C1	Specifies that the list of messages returned should only include messages whose end of their Application TimeInterval (Document TimeInterval) or Server Timestamp comes after the provided date.
EndTime	dateTime	C1	Specifies that the list of messages returned should only include messages whose start of their Application TimeInterval or ServerTimestamp (when the message was received or published in the server) comes before the provided date.
IntervalType	String	C1	Indicates whether the StartTime and EndTime refer to Application TimeInterval or to Server Timestamp. Permitted values: “Application” (default), “Server”
Code	number	C2	Specifies that the list of messages returned should only include messages with an internal identification number higher than the provided code. This means that the list will contain messages that are newer to the given one. For optimization purposes, if this filter is used, only messages available since the

			00.00 of D-1 (day before) are guaranteed to be included in the response list
MessageIdentification	string	O	Specifies that the list of messages returned should only include messages whose Message Identification is compliant with the pattern provided in this parameter. ("*" can be used as a wildcard)
MsgType	string	O	Specifies that the list of messages returned should only include messages of the provided type.
Owner	string	O	Specifies that the list of messages returned should only include messages belonging to the provided Owner.

5.1.3 Service Response

If there is no message according to the provided filters, the service will return an empty list. Otherwise, a list of message descriptors will be returned. Each message descriptor will include the following parameters:

Parameter Name	Type	M/O/C	Description
Code	Position Type (number)	M	Specifies the internal identification number of the message
MessageIdentification	Identification Type (string)	M	Specifies the Message Identification. Messages defined in IEC-62325 Part 451-X series include this information. For additional messages not included in that standard, the server shall have a way of assigning a MessageIdentification to those messages.
MessageVersion	VersionType	O	Specifies the Message Version. Messages defined in IEC-62325 Part 451-X series include this information. For additional messages not included in that standard, the server should have a way of assigning a Message Version to those messages.
Status	String	O	Specifies the status of messages. Corresponds with the main reason code of the Acknowledgement message associated with this message as per IEC-62325-451-1. Possible values are: OK, FAILED. The status value "OK" corresponds with the IEC-62325-451-1 ReasonCode "A01", and the status value "FAILED" corresponds with the rest of ReasonCodes.
ApplicationTimeInterval.Start	dateTime	M	Specifies the start of the message Application Time Interval. Messages defined in IEC-62325 Part 451-X series include this information. For additional messages not included in that standard, the server shall have a way of assigning an Application TimeInterval to those messages
ApplicationTimeInterval.End	dateTime	O	Specifies the end of the message Application Time Interval. Messages defined in IEC-62325 Part 451-X series include this information. For additional messages not included in that standard, the server shall have a way of assigning an Application TimeInterval to those messages.

			When this information is missing, the message Application Time Interval is “from ApplicationTimeIntervalStart on” without an explicit end.
ServerTimestamp	MessageDateTime Type	M	Specifies the server timestamp (when the message was received or published in the server) of the message
Type	LongIdentification Type (string)	M	Specifies the Message Type.
Owner	LongIdentification Type (string)	M	Specifies the Data Owner of the message.

5.1.4 Functional Requirements

Confidentiality rules of the European Energy Market for Electricity shall be observed, thus the list of messages available to a client will only include those messages to which he is entitled (either completely or partially).

A client will be able to see all his previously submitted messages to the server, their responses sent from the server (acknowledgements), and any publications that are available to the client.

When the service is called with an invalid filter (e.g. malformed application dates) a Fault message will be returned.

5.2 Get Message

5.2.1 General

The Get Messages service is used to obtain the message associated to the given parameter (filter).

The filter shall be one of the following:

- Message Identification and Version
- Message code
- Queue indication

5.2.2 Service Request

Parameter Name	Type	M/O/C	Description
MessageIdentification	Identification Type (string)	C1	Specifies the Message Identification of the requested message.
MessageVersion	VersionType	C1	Specifies the Message Version of the requested message. If more than one message in the server have the same MessageIdentification and MessageVersion, the most recent one will be returned.
Code	Position Type (number)	C2	Specifies the internal identification number of the requested message
Queue	String	C3	Indicates that the server will decide which message will be returned. Its value shall be “NEXT”.

5.2.3 Service Response

Parameter Name	Type	M/O/C	Description

[First child of Payload]	Any	C1	The XML message that is being returned to the client.
BinaryContent	Base64Binary	C2	Optionally binary content may also be returned depending on the type of the requested message.
BinaryName	String	C2	Optionally, the name of the requested binary file.

5.2.4 Functional Requirements

Only one message will be retrieved for each Get Message service invocation.

If the retrieved message is a File, then the content is expressed as base 64 encoded wrapped by the optional tag “BinaryContent”.

When the service is called with an invalid message (e.g. missing or invalid code) a Fault message will be returned.

The Queue parameter can be used when the server keeps an ordered list of messages for each client to retrieve. A server not supporting this feature will return a fault message.

5.3 Put Message

5.3.1 General

The Put Message service is used to send a message to the server for further processing following the rules of the European Energy Markets for Electricity.

A series of standard XML messages related to the European Energy Market for Electricity are defined in the IEC-62325 Part 451-X series, but this specification allows servers to process additional XML messages not defined in said series.

Optionally, binary files may also be sent, if supported by the server.

5.3.2 Service Request

Parameter Name	Type	M/O/C	Description
[First child of Payload]	Any	C1	The XML message that is being sent to the server.
BinaryContent	Base64Binary	C2	Optionally binary content may also be sent to the server.
BinaryName	String	C2	Optionally, the name of the binary file sent to the server.

5.3.3 Service Response

The response from the server will be in the form of an XML message indicating the technical and/or functional acceptance or rejection of the message. For the messages described in the IEC-62325 Part 451-X series, the response from the server should be an acknowledgement message as defined in IEC-62325 Part 451-1.

5.3.4 Functional Requirements

For each XML message received the server needs to be able to identify each individual message. The IEC-62325 Part 451-X series define such a way via the elements DocumentIdentification and optionally DocumentVersion.

The server will perform simple validations:

- A user cannot send two (or more) messages with the same identification (e.g. Document Identification and Version). However, another user could send messages with the same identification that other user sent before.

- A user cannot send a message whose Version is lower than another message that has been sent previously.

If there is not enough information in the sent message to create a proper response (incomplete XML or missing tags), a Fault message will be issued instead.

5.4 Query Data

5.4.1 General

The non-mandatory Query Data Service can be used by clients to request specific data from the server using different query parameters.

The server does not need to have the response XML message ready before the service invocation, and can create a specific message in response to each request as needed.

5.4.2 Service Request

Parameter Name	Type	M/O	Description
DataType	String	M	Indicates the type of data being requested.
StartTime	dateTime	O	Specifies that the returned message should only include data whose Application Date is after the provided date.
EndTime	dateTime	O	Specifies that the returned message should only include data whose Application Date is before the provided date.
BD1 … BDn	String	O	Specifies additional parameters for the service.

5.4.3 Service Response

Parameter Name	Type	M/O/C	Description
QueryData	gop:QueryData	M	The XML message that is being returned to the client.

5.4.4 Functional Requirements

Only one message will be retrieved for each Query Data service invocation.

When the service is called with invalid parameters (e.g. invalid AreaCode) a Fault message will be returned.

A list of basic datatypes to be supported if this service is implemented is shown below:

DataType value	M/O	Description
"listOfDataTypes"	M	The server will return a list of valid DataTypes that can be used for this service on this server.
"serverTimestamp"	M	The server will return the Server Timestamp in UTC format
"parameterLimits"	O	The server will return the server limits for parameters used in the List, Get, Put services, such as maximum message size, maximum number of queried days in the list service…

6 Applying IEC 61968-100

6.1 Integration Pattern

6.1.1 General

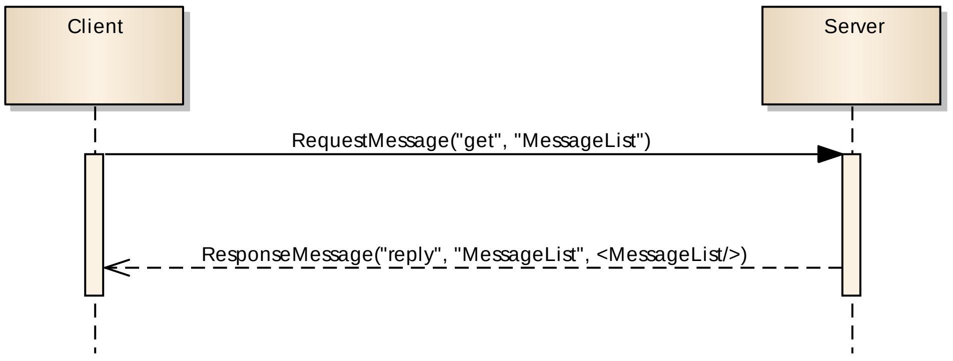
The interactions between Client and Server described in this specification correspond with the IEC 61968-100 use case “Simple Request/Reply” supported by the “Basic Web Service Integration Pattern”.

This does not preclude the implementation of other Integration Patterns to support additional use cases such us “Request/Reply with ESB”, “Events” etc.

The server will expose in a WSDL a single operation “request” that will serve as the single entry point for the services described in this document.

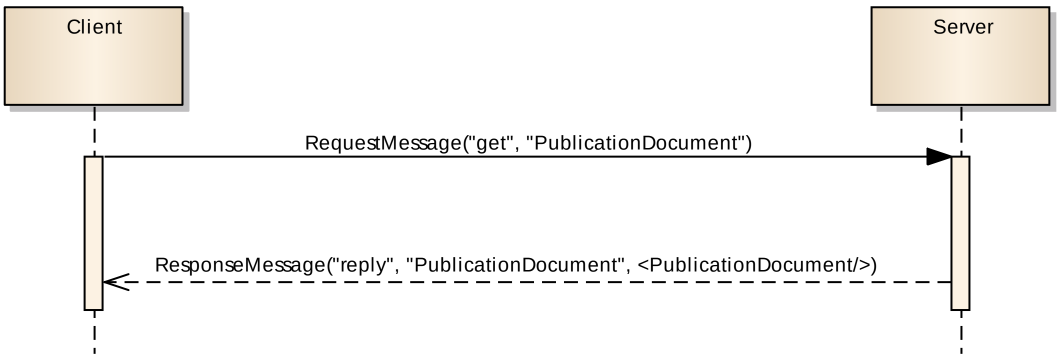
In the following diagrams the request/reply exchange between client and server is shown, including the type of message sent, the IEC 61968-100 verb and noun used, and the payload when applicable.

6.1.2 List Service


Client
Server
RequestMessage("get", "MessageList")
ResponseMessage("reply", "MessageList", <MessageList/>)

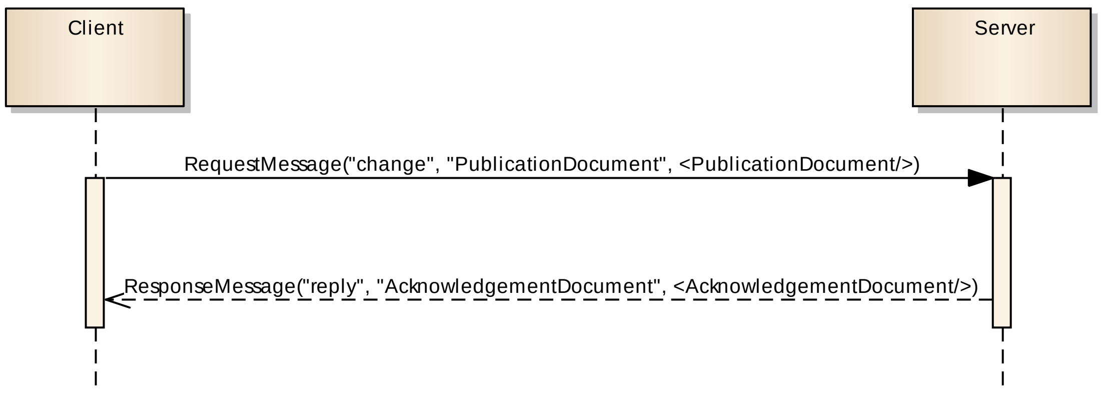
6.1.3 Get Service

In this example, a message of type “PublicationDocument” is requested:


Client
Server
RequestMessage("get", "PublicationDocument")
ResponseMessage("reply", "PublicationDocument", <PublicationDocument/>)

6.1.4 Put Service

In this example, a message of type “PublicationDocument” is sent to the server:


Client
Server
RequestMessage("change", "PublicationDocument", <PublicationDocument/>)
ResponseMessage("reply", "AcknowledgementDocument", <AcknowledgementDocument/>)

6.2 Service Mapping

6.2.1 General

The mapping between the different services described in this document and the IEC 61968-100 messages, including some relevant parameters, is shown in the following table:

Service	61968-100 Stereotype Msg.	61968-100 Verb	61968-100 Noun	61968-100 Payload
List (Req.)	RequestMessage	get	MessageList	n/a
List (Resp)	ResponseMessage	reply	MessageList	gop:MessageList
Get (Req)	RequestMessage	get	Any	n/a
Get (Resp)	ResponseMessage	reply	*ChildOfPayload*	Any
Put (Req.)	RequestMessage	create	*ChildOfPayload*	Any
Put (Resp)	ResponseMessage	reply	*ChildOfPayload (Ack)*	Any
QueryData (Req.)	RequestMessage	get	QueryData	n/a
QueryData (Resp.)	ResponseMessage	reply	QueryData	gop:QueryData

Unless otherwise stated, all elements in the Header, Request or Response defined in IEC61968-100 can be used.

6.2.2 Header Values

The “Source” element will be the EIC code of the Sender Entity for messages following IEC 62325-451-n series.

6.2.3 Request Values

The parameters in the List Service Request, Get Service Request and QueryData Service Request that do not correspond directly with a parameter in the IEC 61968-100 msg:RequestMessage will be provided as Name/Value pairs under the msg:Option element of the msg:Request.

In the case of the Put service Request, when sending a binary file, the name of the file will be included in the msg:Request.ID, with the attribute idType equal to “name”.

6.2.4 Response Values

In the case of the Put service Response, the msg:Reply.Result values “OK” and “FAILED” correspond with the “Completely Accepted” and “Completely Rejected” responses described in IEC-62325 Part 451-1.

In the case of the Get service Response, when requesting a binary file, the name of the file will be included in the msg:Response.ID, with the attribute idType equal to “name”.

6.2.5 Payload Values

The generic message payload container will be used (it is not mandatory to have strongly typed WSDLs)

The “BinaryContent” described in this specification will be mapped to the “msg:Compressed” element of the EIC 61968-100.

The “msg:Format” element will be absent or have the value “XML” if the payload is an XML, and “BINARY” if the payload is a binary document.

7 Schema Definitions

7.1 Common Definitions

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:iec62325.504:messages:1:0" targetNamespace="urn:iec62325.504:messages:1:0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:annotation>
    <xsd:documentation>TF EDI EME - Common Types</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType name="TimeIntervalType">
    <xsd:sequence>
      <xsd:element name="start" type="xsd:dateTime"/>
      <xsd:element name="end" type="xsd:dateTime" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="StatusType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="OK"/>
      <xsd:enumeration value="FAILED"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

7.2 List Message

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:iec62325.504:messages:1:0" targetNamespace="urn:iec62325.504:messages:1:0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:annotation>
    <xsd:documentation>TF EDI EME - List</xsd:documentation>
  </xsd:annotation>
  <xsd:element name="MessageList">
    <xsd:annotation>
      <xsd:documentation>List of messages</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Message" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Code" type="xsd:positiveInteger"/>
              <xsd:element name="MessageIdentification" type="xsd:string"/>
              <xsd:element name="MessageVersion" type="xsd:positiveInteger"
minOccurs="0"/>
              <xsd:element name="Status" type="StatusType" minOccurs="0"/>
              <xsd:element name="ApplicationTimeInterval" type="TimeIntervalType"/>
              <xsd:element name="ServerTimestamp" type="xsd:dateTime"/>
              <xsd:element name="Type" type="xsd:string"/>
              <xsd:element name="Owner" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

7.3 QueryData Message

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:iec62325.504:messages:1:0" targetNamespace="urn:iec62325.504:messages:1:0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:annotation>
    <xsd:documentation>TF EDI EME - QueryData</xsd:documentation>
  </xsd:annotation>
  <xsd:element name="QueryData">
    <xsd:annotation>
      <xsd:documentation>Wraps the response from the "QueryData"
service</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence minOccurs="0">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

8 Service Provider WSDL Abstract Definitions

By using synchronous web services, a client is immediately aware of the result of the service invocation.

This clause specifies the abstract WSDL definitions to support the services specified in this document.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:cmsg="urn:iec62325.504:messages:1:0" xmlns:wss="urn:iec62325.504:wss:1:0"
xmlns:msg="http://iec.ch/TC57/2011/schema/message"
targetNamespace="urn:iec62325.504:wss:1:0">
    <types>
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:import namespace="urn:iec62325.504:messages:1:0" schemaLocation="./urn-
iec62325-504-messages-1-0.xsd"/>
            <xs:import namespace="http://iec.ch/TC57/2011/schema/message"
schemaLocation="./http-iec-ch-TC57-2011-schema-message.xsd"/>
        </xs:schema>
    </types>
    <message name="msgRequestMessage">
        <part name="parameter" element="msg:RequestMessage"/>
    </message>
    <message name="msgResponseMessage">
        <part name="parameter" element="msg:ResponseMessage"/>
    </message>
    <message name="msgFaultMsg">
        <part name="msgFaultMessage" element="msg:FaultMessage"/>
    </message>
    <portType name="port_TFEDI_type">
        <operation name="request">
            <input message="wss:msgRequestMessage"/>
            <output message="wss:msgResponseMessage"/>
            <fault name="msgFaultMessage" message="wss:msgFaultMsg"/>
        </operation>
    </portType>
</definitions>
```

9 Service Provider WSDL SOAP Binding

This clause specifies the binding template of the abstract WSDL definitions in Clause 8 with SOAP Messages with the http(s) transport protocol. Although SOAP 1.2 is shown here, SOAP 1.1 may also be used.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:cmsg="urn:iec62325.504:messages:1:0" xmlns:wss="urn:iec62325.504:wss:1:0"
xmlns:msg="http://iec.ch/TC57/2011/schema/message"
targetNamespace="urn:iec62325.504:wss:1:0">
    <binding name="binding_TFEDI" type="wss:port_TFEDI_type">
        <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="request">
            <soap12:operation soapActionRequired="false" style="document"/>
            <input>
                <soap12:body use="literal"/>
            </input>
            <output>
                <soap12:body use="literal"/>
            </output>
            <fault name="msgFaultMessage">
                <soap12:fault name="msgFaultMessage" use="literal"/>
            </fault>
        </operation>
    </binding>
    <service name="ServiceEME">
        <port name="Service_EME_Port" binding="wss:binding_TFEDI">
            <soap12:address location="https://example.com/WebService_EME/Service_EME"/>
        </port>
    </service>
</definitions>
```

Fault messages are used when the application produces an error condition that cannot be expressed with the operation response. Those errors are not usually related to business rules.

The fault message consists of two parts, a code and reason (also known as a fault string) and, optionally a detailed block that can be used to include more information about the error.

10 Security

The confidentiality of communications, and therefore the confidentiality of the information transmitted are achieved in the transport layer using https with client and server certificates.

To ensure the authentication of the signer, the document integrity and non-repudiation, all IEC 61968-100 messages with XML documents in Payloads for the “Get”, “Put” and “QueryData” services are digitally signed either by the Data Owner or the Data Provider.

This digital signature, made with the private key associated with the user's digital certificate, follows the "XML Digital Signature" standard, and is transmitted in the signed message (enveloped signature). The signature is included in the IEC 61968-100 messages Header.

The signature applies to the complete IEC 61968-100 message that is being sent or retrieved, not to the XML Soap, nor exclusively to the single XML document in the Payload.

Thus the Reference URI shall always be “” in order to point to the whole IEC 61968-100 document.

The CanonicalizationMethod will be one of:

- http://www.w3.org/TR/2001/REC-xml-c14n-20010315
- http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
- http://www.w3.org/2001/10/xml-exc-c14n
- http://www.w3.org/2001/10/xml-exc-c14n#WithComments

The set of supported transforms (in this order) are:

- http://www.w3.org/2000/09/xmldsig#enveloped-signature
- (optional) any of the permitted canonicalization algorithms

In particular, no XPath or XSLT transforms are allowed.

The element “KeyInfo” includes the public key of the certificate used to create the signature.

In the case of messages with binary content, document signing is similar; since the “Compressed” element falls inside the IEC 61968-100 message that is being signed

ANNEX I: Message Examples

1 List

1.1 Basic Example

1.1.1 Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
…
   <soap:Body>
      <msg:RequestMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
         <msg:Header>
            <msg:Verb>get</msg:Verb>
            <msg:Noun>MessageList</msg:Noun>
            <msg:Context>PRODUCTION</msg:Context>
            <msg:AckRequired>true</msg:AckRequired>
         </msg:Header>
         <msg:Request>
            <msg:StartTime>2012-11-26T23:00:00Z</msg:StartTime>
            <msg:EndTime>2012-11-27T23:00:00Z</msg:EndTime>
            <msg:Option>
               <msg:name>IntervalType</msg:name>
               <msg:value>Server</msg:value>
            </msg:Option>
            <msg:Option>
               <msg:name>Owner</msg:name>
               <msg:value>10X1001A1001A450</msg:value>
            </msg:Option>
         </msg:Request>
      </msg:RequestMessage>
   </soap:Body>
</soap:Envelope>
```

1.1.2 Response:

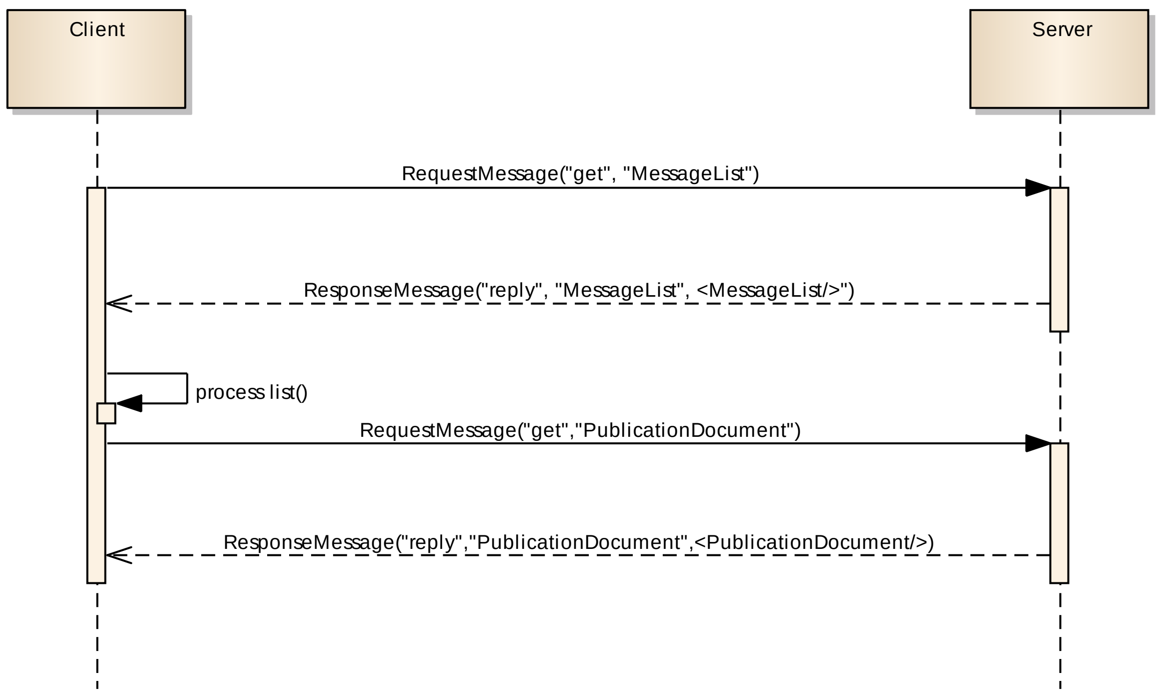
```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header/>
   <soap:Body>
      <msg:ResponseMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
         <msg:Header>
            <msg:Verb>reply</msg:Verb>
            <msg:Noun>MessageList</msg:Noun>
            <msg:Context>PRODUCTION</msg:Context>
         </msg:Header>
         <msg:Reply>
            <msg:Result>OK</msg:Result>
         </msg:Reply>
         <msg:Payload>
            <urn:MessageList xmlns:urn="urn:iec62325.504:messages:1:0">
               <urn:Message>
                  <urn:Code>879021</urn:Code>
                  <urn:MessageIdentification>SOAM-TSO-TSO-
301012</urn:MessageIdentification>
                  <urn:MessageVersion>2</urn:MessageVersion>
                  <urn:Status>OK</urn:Status>
                  <urn:ApplicationTimeInterval>
                     <urn:start>2012-11-26T23:00:00Z</urn:start>
                     <urn:end>2012-11-27T23:00:00Z</urn:end>
                  </urn:ApplicationTimeInterval>
                  <urn:ServerTimestamp>2012-11-
25T07:14:23Z</urn:ServerTimestamp>
                  <urn:Type>SOAM</urn:Type>
                  <urn:Owner>10XIC-TSO------E</urn:Owner>
               </urn:Message>
            </urn:MessageList>
```

```
          </msg:Payload>
       </msg:ResponseMessage>
    </soap:Body>
</soap:Envelope>
```

2 Get

As a general rule, before retrieving a Message, the client should find out if the message exists.

Thus, the recommended sequence diagram is the following figure, where the actor requests the ”List” service in first place, recovering the list of available messages. Once the actor has such information, he will invoke the “Get” service for a message that appears in the retrieved list.


Client
Server
RequestMessage("get", "MessageList")
ResponseMessage("reply", "MessageList", <MessageList/>")
process list()
RequestMessage("get","PublicationDocument")
ResponseMessage("reply","PublicationDocument",<PublicationDocument/>)

2.1 Basic Example

2.1.1 Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header/>
    <soap:Body>
        <msg:RequestMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
            <msg:Header>
                <msg:Verb>get</msg:Verb>
                <msg:Noun>Schedule_MarketDocument</msg:Noun>
                <msg:Context>PRODUCTION</msg:Context>
                <msg:AckRequired>true</msg:AckRequired>
            </msg:Header>
            <msg:Request>
                <msg:Option>
                    <msg:name>Code</msg:name>
                    <msg:value>879021</msg:value>
```

```
                 </msg:Option>
             </msg:Request>
         </msg:RequestMessage>
     </soap:Body></soap:Envelope>
```

2.1.2 Response:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header>
   </soap:Header>
   <soap:Body>
       <msg:ResponseMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
            <msg:Header>
                <msg:Verb>reply</msg:Verb>
                <msg:Noun>Schedule_MarketDocument</msg:Noun>
                <msg:Context>PRODUCTION</msg:Context>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
         <SignedInfo>
            <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <Reference URI="">
               <Transforms>
                  <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                  <Transform Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
               </Transforms>
               <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
               <DigestValue>xSnE7qGXJkWGG11rq3ze5DYGxHw=</DigestValue>
            </Reference>
         </SignedInfo>
<SignatureValue>QLk/r49g4uoR8pXT9WX3fg6QYEh/r0……==</SignatureValue>
         <KeyInfo>
            <X509Data>
               <X509IssuerSerial>
                  <X509IssuerName>CN=testCA, O=test
Domain</X509IssuerName>
                  <X509SerialNumber>238</X509SerialNumber>
               </X509IssuerSerial>
               <X509SubjectName>UID=myname, OU=intern, OU=myou, OU=users,
O=test, C=test</X509SubjectName>
               <X509Certificate>MIID2TCCAsGgAwIBA ....
=</X509Certificate>
            </X509Data>
         </KeyInfo>
      </Signature>
            </msg:Header>
            <msg:Reply>
                <msg:Result>OK</msg:Result>
            </msg:Reply>
            <msg:Payload>
<Schedule_MarketDocument xmlns="urn:iec62325.351:tc57wg16:451-
2:scheduledocument:5:0">
   <mRID>20140416_CAS_D_RTE_REE</mRID>
   <revisionNumber>1</revisionNumber>
   <type>A04</type>
   <process.processType>A01</process.processType>
   <process.classificationType>A01</process.classificationType>
   <sender_MarketParticipant.mRID codingScheme="A01">10XFR-RTE------Q
</sender_MarketParticipant.mRID>
   <sender_MarketParticipant.marketRole.type>A04
</sender_MarketParticipant.marketRole.type>
```

```
   <receiver_MarketParticipant.mRID codingScheme="A01">10XES-REE------E
</receiver_MarketParticipant.mRID>

<receiver_MarketParticipant.marketRole.type>A04</receiver_MarketParticipa
nt.marketRole.type>
   <createdDateTime>2014-04-15T13:06:29Z</createdDateTime>
   <schedule_Time_Period.timeInterval>
      <start>2014-04-15T22:00Z</start>
      <end>2014-04-16T22:00Z</end>
   </schedule_Time_Period.timeInterval>
   <domain.mRID codingScheme="A01">10YDOM--ES-FR--D</domain.mRID>
   <matching_Time_Period.timeInterval>
      <start>2014-04-15T22:00Z</start>
      <end>2014-04-16T22:00Z</end>
   </matching_Time_Period.timeInterval>
...
</Schedule_MarketDocument>
            </msg:Payload>
        </msg:ResponseMessage>   </soap:Body>
</soap:Envelope>
```

3 Put

3.1 Basic Example

3.1.1 Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header>
   </soap:Header>
    <soap:Body>
       <msg:RequestMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
           <msg:Header>
               <msg:Verb>create</msg:Verb>
               <msg:Noun>Schedule_MarketDocument</msg:Noun>
               <msg:Context>PRODUCTION</msg:Context>
               <msg:AckRequired>true</msg:AckRequired>
     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
     ...
     </Signature>
         </msg:Header>
         <msg:Payload>
<Schedule_MarketDocument xmlns="urn:iec62325.351:tc57wg16:451-
2:scheduledocument:5:0">
   <mRID>20140416_CAS_D_RTE_REE</mRID>
   <revisionNumber>1</revisionNumber>
   <type>A04</type>
   <process.processType>A01</process.processType>
   <process.classificationType>A01</process.classificationType>
   <sender_MarketParticipant.mRID codingScheme="A01">10XFR-RTE------Q
</sender_MarketParticipant.mRID>
   <sender_MarketParticipant.marketRole.type>A04
</sender_MarketParticipant.marketRole.type>
   <receiver_MarketParticipant.mRID codingScheme="A01">10XES-REE------E
</receiver_MarketParticipant.mRID>
   <receiver_MarketParticipant.marketRole.type>A04
</receiver_MarketParticipant.marketRole.type>
   <createdDateTime>2014-04-15T13:06:29Z</createdDateTime>
...
</Schedule_MarketDocument>
         </msg:Payload>
      </msg:RequestMessage>
   </soap:Body></soap:Envelope>
```

3.1.2 Response:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header>
   </soap:Header>
    <soap:Body>
       <msg:ResponseMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
            <msg:Header>
                <msg:Verb>reply</msg:Verb>
                <msg:Noun>Acknowledgement_MarketDocument</msg:Noun>
                <msg:Context>PRODUCTION</msg:Context>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      ...
      </Signature>
            </msg:Header>
            <msg:Reply>
                <msg:Result>OK</msg:Result>
            </msg:Reply>
            <msg:Payload>
<Acknowledgement_MarketDocument xmlns="urn:iec62325.351:tc57wg16:451-
1:acknowledgementdocument:6:0">
   <mRID>ACK_ASFR-ES-I2-HOURLY-140415-01</mRID>
   <createdDateTime>2014-04-15T08:55:30Z</createdDateTime>
   <sender_MarketParticipant.mRID codingScheme="A01">10XFR-RTE------Q
</sender_MarketParticipant.mRID>
   <sender_MarketParticipant.marketRole.type>A04
</sender_MarketParticipant.marketRole.type>
   <receiver_MarketParticipant.mRID codingScheme="A01">10XES-REE------E
</receiver_MarketParticipant.mRID>
   <receiver_MarketParticipant.marketRole.type>A04
</receiver_MarketParticipant.marketRole.type>
   <received_MarketDocument.mRID>ASFR-ES-I2-HOURLY-----140415-01
</received_MarketDocument.mRID>
   <received_MarketDocument.revisionNumber>1
</received_MarketDocument.revisionNumber>
   <received_MarketDocument.type>A51</received_MarketDocument.type>
   <received_MarketDocument.createdDateTime>2014-04-15T08:55:30Z
</received_MarketDocument.createdDateTime>
   <Reason>
      <code>A01</code>
   </Reason>
</Acknowledgement_MarketDocument>
            </msg:Payload>
        </msg:ResponseMessage>
    </soap:Body>
</soap:Envelope>
```

3.2 Example with binary data

If the message has a binary part, this can be transmitted as 64b encoded form:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header>… …</soap:Header>
    <soap:Body>
        <msg:RequestMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
            <msg:Header>
                <msg:Verb>create</msg:Verb>
                <msg:Noun>Schedule_MarketDocument_pdf</msg:Noun>
                <msg:Context>PRODUCTION</msg:Context>
                <msg:AckRequired>true</msg:AckRequired>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      ...
      </Signature>
            </msg:Header>
            <msg:Request>
               <msg:ID idType="name">schedule_xyz.pdf</msg:ID>
            </msg:Request>
            <msg:Payload>
                <msg:Compressed>
PFNjaGVkdWxlX01hcmtldERvY3VtZW50IHhtbG5zPSJ1cm46aWVjNjIzMjUuMzUxOnRjNTd3Z
zE2OjQ1MS0yOnNjaGVkdWxlZG9jdW1lbnQ6NTowIj4NCiAgIDxtUklEPjIwMTQwNDE2X0NBU1
9EX1JURV9SRUU8L21SSUQ+DQogICA8cmV2aXNpb25OdW1iZXI+MTwvcmV2aXNpb25OdW1iZXI
+DQogICA8dHlwZT5BMDQ8L3R5cGU+DQogICA8cHJvY2Vzcy5wcm9jZXNzVHlwZT5BMDE8L3By
b2Nlc3MucHJvY2Vzc1R5cGU+DQogICA8cHJvY2Vzcy5jbGFzc2lmaWNhdGlvblR5cGU+QTAxP
C9wcm9jZXNzLmNsYXNzaWZpY2F0aW9uVHlwZT4NCiAgIDxzZW5kZXJfTWFya2V0UGFydGljaX
BhbnQubVJJRCBjb2RpbmdTY2hlbWU9IkEwMSI+MTBYRlItUlRFLS0tLS0tUTwvc2VuZGVyX01
hcmtldFBhcnRpY2lwYW50Lm1SSUQ+DQogICA8c2VuZGVyX01hcmtldFBhcnRpY2lwYW50Lm1h
cmtldFJvbGUudHlwZT5BMDQ8L3NlbmRlcl9NYXJrZXRQYXJ0aWNpcGFudC5tYXJrZXRSb2xlL
nR5cGU+DQogICA8cmVjZWl2ZXJfTWFya2V0UGFydGljaXBhbnQubVJJRCBjb2RpbmdTY2hlbW
U9IkEwMSI+MTBYRVMtUkVFLS0tLS0tRTwvcmVjZWl2ZXJfTWFya2V0UGFydGljaXBhbnQubVJ
JRD4NCi4uLg0KPC9TY2hlZHVsZV9NYXJrZXREb2N1bWVudD4= </msg:Compressed>
                <msg:Format>BINARY</msg:Format>
            </msg:Payload>
        </msg:RequestMessage>
    </soap:Body></soap:Envelope>
```

4 Query Data

4.1 List of Data Types Example

4.1.1 Request:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
...
   <soap:Body>
      <msg:RequestMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
         <msg:Header>
            <msg:Verb>get</msg:Verb>
            <msg:Noun>QueryData</msg:Noun>
            <msg:Context>PRODUCTION</msg:Context>
         </msg:Header>
         <msg:Request>
            <msg:Option>
               <msg:name>DataType</msg:name>
               <msg:value>listOfDataTypes</msg:value>
            </msg:Option>
         </msg:Request>
      </msg:RequestMessage>
   </soap:Body>
</soap:Envelope>
```

4.1.2 Response:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header/>
   <soap:Body>
      <msg:ResponseMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
         <msg:Header>
            <msg:Verb>reply</msg:Verb>
            <msg:Noun>QueryData</msg:Noun>
            <msg:Context>PRODUCTION</msg:Context>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      ...
      </Signature>
```

```
         </msg:Header>
         <msg:Reply>
            <msg:Result>OK</msg:Result>
         </msg:Reply>
         <msg:Payload>
            <QueryData xmlns="urn:iec62325.504:messages:1:0">
               <ListOfDataTypes>
                  <DataType>listOfDataTypes</DataType>
                  <DataType>serverTimestamp</DataType>
                  <DataType>parameterLimits</DataType>
               </ListOfDataTypes>
            </QueryData>
         </msg:Payload>
      </msg:ResponseMessage>
   </soap:Body>
</soap:Envelope>
```

5 Fault

5.1 SOAP 1.2

The following shows a fault message returned by a “List” operation where the given TimeInterval is not valid, using SOAP 1.2:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <soap:Fault xmlns:ns4="http://schemas.xmlsoap.org/soap/envelope/">
         <soap:Code>
            <soap:Value>S:Receiver</soap:Value>
         </soap:Code>
         <soap:Reason>
            <soap:Text xml:lang="es">EPF-04</S:Text>
         </soap:Reason>
         <soap:Detail>            <msg:FaultMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
                <msg:Reply>
                    <msg:Result>FAILED</msg:Result>
                    <msg:Error>
                        <msg:code>EPF-04</msg:code>
                        <msg:details>Invalid Time Interval: 2011-
03aT22:00Z2011-03-02T12:00Z </msg:details>
                    </msg:Error>
                </msg:Reply>
            </msg:FaultMessage>      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

5.2 SOAP 1.1

The following shows a fault message returned by a “List” operation where the given TimeInterval is not valid, using SOAP 1.1:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <soap:Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>soap:soaperver</faultcode>
         <faultstring>EPF-04</faultstring>
         <detail>

            <msg:FaultMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
```

```
                <msg:Reply>
                    <msg:Result>FAILED</msg:Result>
                    <msg:Error>
                        <msg:code>EPF-04</msg:code>
                        <msg:details>Invalid Time Interval: 2011-
03aT22:00Z2011-03-02T12:00Z </msg:details>
                    </msg:Error>
                </msg:Reply>
            </msg:FaultMessage>
         </detail>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

6 Digital Signature

6.1 Basic Example

This is an example of a response to the Put service:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://www.w3.org/2003/05/soap-envelope">
   <SOAP-ENV:Header>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
       <msg:ResponseMessage
xmlns:msg="http://iec.ch/TC57/2011/schema/message">
            <msg:Header>
                <msg:Verb>reply</msg:Verb>
                <msg:Noun>Acknowledgement_MarketDocument</msg:Noun>
                <msg:Context>PRODUCTION</msg:Context>
         <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
            <SignedInfo>
               <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
               <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></SignatureMethod>
               <Reference URI="">
                  <Transforms>
                     <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                     <Transform Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"></Transform>
                  </Transforms>
                  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <DigestValue>g87DHbL2vOP4N4sCUBjby/M6K48=</DigestValue>
               </Reference>
            </SignedInfo>

<SignatureValue>sl3aQ+UsBzesV6EQfGidB7Iuyk/iLIQP1VlKtyOEUEphGfP3y7no2tV9t
w+mH2FmId90hKCSurwnkakpKSryUNUDI9i5NPz+g+sI4N/hz0rVsSfh/CZV1ZaboVdVbngv/y
c7JE77NFwRe6kOAmWnUvnSywx162ZOasuEXxc+olkD+7HtPzAR86JhB2oAqqdtcuc+FGuMLns
Ci6p5whZRq99rkhvxnup3w2Y6b84nlJm/FJD9RsZJ0UInXVLMgALUmbxClKlKXwFUYLjSqY/S
rFjFzkm0KshgWiFKOYhwGhUy0+UNCpfRcGncOZj61tIuYD8bT205JITgSkdHO4bRmw==</Sig
natureValue>
            <KeyInfo>
               <X509Data>
                  <X509IssuerSerial>
                     <X509IssuerName>CN=testCA, O=test
Domain</X509IssuerName>
                     <X509SerialNumber>131</X509SerialNumber>
                  </X509IssuerSerial>
                  <X509SubjectName> UID=myname, OU=intern, OU=myou,
OU=users, O=test, C=test</X509SubjectName>
                  <X509Certificate>
MIICOTCCAaKgAwIBAAIEU04yjTANBgkqhkiG9w0BAQUFADBhMQ0wCwYDVQQIEwRL
RVkhMQ8wDQYDVQQHEwZTRUNSRVQxDDAKBgNVBAoTA1RIRTEXMBUGA1UECxMOWU9V
IEhBVkUgRk9VTkQxGDAWBgNVBAMTD0NPTkdSQVRVTEFUSU9OUzAeFw0xNDA0MTYw
NzM0MzdaFw0xNTA4MjkwNzM0MzdaMGExDTALBgNVBAgTBEtFWSExDzANBgNVBAcT
BlNFQ1JFVDEMMAoGA1UEChMDVEhFMRcwFQYDVQQLEw5ZT1UgSEFWRSBGT1VORDEY
MBYGA1UEAxMPQ09OR1JBVFVMQVRJT05TMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQCQSgndj2T9o19c1anXRkLeiVnFc8K1BhthMddZQ6kMZWcrFhKdBmrA0Fok
JLYttpkVLcYHQcSP3+MygBItFpkhhoTPhPT2IjN4899cDGw10zG4fHjJF3Vn3S4v
+TiUKilb0yfVHPl2KqSgLd2X49yaNoc+97Rp2adYPuWxnLNHhwIDAQABMA0GCSqG
SIb3DQEBBQUAA4GBAAn55WsZTpnRBkDhP9EwsMAqOf+4nGzYn+zqeKrczico8ylj
9ndLFl71aFMxQLVd8u8EaW8kGbo92Chh4z4vm9y4AFCFeHM5EwBA8w9tOw5l7oU9
+NmMg6owGY6m+vP31623C/lbv66dQOHUXLO5H/VVTOUgBOyoGXBtaC1sGvm8
</X509Certificate>
               </X509Data>
            </KeyInfo>
         </Signature>

            </msg:Header>
            <msg:Reply>
                <msg:Result>OK</msg:Result>
            </msg:Reply>
            <msg:Payload>
<Acknowledgement_MarketDocument xmlns="urn:iec62325.351:tc57wg16:451-
1:acknowledgementdocument:6:0">
   <mRID>ACK_ASFR-ES-I2-HOURLY-140415-01</mRID>
   <createdDateTime>2014-04-15T08:55:30Z</createdDateTime>
   <sender_MarketParticipant.mRID codingScheme="A01">10XFR-RTE------Q
</sender_MarketParticipant.mRID>
   <sender_MarketParticipant.marketRole.type>A04
</sender_MarketParticipant.marketRole.type>
   <receiver_MarketParticipant.mRID codingScheme="A01">10XES-REE------E
</receiver_MarketParticipant.mRID>
   <receiver_MarketParticipant.marketRole.type>A04
</receiver_MarketParticipant.marketRole.type>
   <received_MarketDocument.mRID>ASFR-ES-I2-HOURLY-----140415-01
</received_MarketDocument.mRID>
   <received_MarketDocument.revisionNumber>1
</received_MarketDocument.revisionNumber>
   <received_MarketDocument.type>A51</received_MarketDocument.type>
   <received_MarketDocument.createdDateTime>2014-04-15T08:55:30Z
</received_MarketDocument.createdDateTime>
   <Reason>
      <code>A01</code>
   </Reason>
</Acknowledgement_MarketDocument>
            </msg:Payload>
        </msg:ResponseMessage>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

ANNEX II: Java Code Examples

1 Sign

The following simplified code excerpt signs a message following the rules described in this document:

```
/**
* Signs the given xml document using the given private key and
certificate.
* @param msgAsDocument The document to be signed, the result of the
process
*   will be returned in this parameter.
* @param privateKey The private key to be used for signature.
* @param cert The certificate to be used for signature.
* @throws Exception If unable to sign the document.
* @see #signString(StringBuilder, RSAPrivateKey, X509Certificate)
*/
public static void signDocument(final Document msgAsDocument, final
RSAPrivateKey privateKey, final X509Certificate cert) throws Exception {

    try {
        XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");

        List<Transform> trfLst = new ArrayList<>();
            trfLst.add(fac.newTransform(Transform.ENVELOPED,
(TransformParameterSpec) null));

trfLst.add(fac.newCanonicalizationMethod(CanonicalizationMethod.INCLUSIVE
, (C14NMethodParameterSpec) null));

        Reference ref = fac
                .newReference("", fac.newDigestMethod(DigestMethod.SHA1,
null), trfLst, null, null);

        SignedInfo si =
fac.newSignedInfo(fac.newCanonicalizationMethod(CanonicalizationMethod.IN
CLUSIVE, (C14NMethodParameterSpec) null),
fac.newSignatureMethod(SignatureMethod.RSA_SHA1, null),
                Collections.singletonList(ref));

        Node headerNode = null;
        NodeList nl =
msgAsDocument.getElementsByTagNameNS("http://iec.ch/TC57/2011/schema/mess
age", "Header");
        if (nl.getLength() == 1) {
            headerNode = nl.item(0);
        } else {
            throw new Exception("Invalid document.");
       }

        DOMSignContext dsc = new DOMSignContext(privateKey, headerNode);

        KeyInfoFactory keyInfoFactory = fac.getKeyInfoFactory();
        List<Object> x509Content = new ArrayList<>();

x509Content.add(keyInfoFactory.newX509IssuerSerial(cert.getIssuerDN().get
Name(), cert.getSerialNumber()));
        x509Content.add(cert.getSubjectX500Principal().getName());
        x509Content.add(cert);
        X509Data xd = keyInfoFactory.newX509Data(x509Content);

        KeyInfo keyInfo =
keyInfoFactory.newKeyInfo(Collections.singletonList(xd));

        XMLSignature signature = fac.newXMLSignature(si, keyInfo);
        signature.sign(dsc);

    } catch (Exception e) {  //Simplified exception handling for brevity
        throw e;
    }
}
```

2 Verify

The following simplified code excerpt verifies a message's signature following the rules described in this document

```
/**
 * Verifies the signature of the given signed document.
 * @param msgAsDocument The document to be validated.
 * @throws Exception If the document cannot be validated
 *    or if its signature is invalid.
 */
public static void verifyDocument(final Document msgAsDocument) throws
Exception {

    try {
        Node signatureNode = null;
        NodeList nl =
msgAsDocument.getElementsByTagNameNS(XMLSignature.XMLNS, "Signature");
        if (nl.getLength() == 1) {
            signatureNode = nl.item(0);
        } else {
            throw new Exception("Invalid document.");
        }

        XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");
        DOMValidateContext valContext = new DOMValidateContext(new
KeyValueKeySelector(), signatureNode);
        XMLSignature signature = fac.unmarshalXMLSignature(valContext);
        boolean coreValidity = signature.validate(valContext);

        if (coreValidity == false) {
            StringBuilder sb = new StringBuilder("Signature validation
failed. Signature status: ");

sb.append(signature.getSignatureValue().validate(valContext));
            sb.append(". ");
            Iterator<?> i =
signature.getSignedInfo().getReferences().iterator();
            for (int j = 0; i.hasNext(); j++) {
                boolean refValid = ((Reference)
i.next()).validate(valContext);
                sb.append("ref[").append(j).append("] validity status:
").append(refValid);
            }
            throw new Exception(sb.toString());
        }
    } catch (Exception e) {  //Simplified exception handling for brevity
        throw e;
    }
}
```