



|  |
| --- |
| X-Road: Message Protocol v4.0  **Technical Specification** |
| Version: 4.0.12  02.12.2015  29 pages  Doc. ID: PR-MESS |

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 04.09.2015 | 4.0.2 | Converted to ODT | Siim Annuk |
| 08.09.2015 | 4.0.3 | Minor fixes | Siim Annuk |
| 10.09.2015 | 4.0.4 | Fixed some typos | Siim Annuk |
| 16.09.2015 | 4.0.5 | Editorial changes made | Imbi Nõgisto |
| 30.09.2015 | 4.0.6 | Additional information added about requestHash header field and HTTP headers | Siim Annuk |
| 14.10.2015 | 4.0.7 | Note added about supported attachment encodings. Updated examples | Siim Annuk,  Ilja Kromonov |
| 17.10.2015 | 4.0.8 | Clarified must/MUST language | Margus Freudenthal |
| 28.10.2015 | 4.0.9 | Better example messages added | Siim Annuk |
| 28.10.2015 | 4.0.10 | Complete X-Road identifiers schema added | Siim Annuk |
| 20.11.2015 | 4.0.11 | Minor enhancements, example messages fixed | Siim Annuk |
| 02.12.2015 | 4.0.12 | Minor fixes added | Siim Annuk |

Table of Contents

[License 4](#_Toc437261032)

[1 Introduction 5](#_Toc437261033)

[1.1 Terms and Abbreviations 5](#_Toc437261034)

[1.2 References 5](#_Toc437261035)

[1.3 Identifying Entities 5](#_Toc437261036)

[2 Format of Messages 7](#_Toc437261037)

[2.1 Identifiers 7](#_Toc437261038)

[2.2 Message Headers 8](#_Toc437261039)

[2.3 Message Body 10](#_Toc437261040)

[2.4 Attachments 10](#_Toc437261041)

[2.5 Fault Messages 10](#_Toc437261042)

[3 Describing Services 11](#_Toc437261043)

[3.1 General 11](#_Toc437261044)

[3.2 Describing Services with WSDL 11](#_Toc437261045)

[Annex A XML Schema for Identifiers 13](#_Toc437261046)

[Annex B XML Schema for Messages 16](#_Toc437261047)

[Annex C Example WSDL 18](#_Toc437261048)

[Annex D Example Fault Messages 24](#_Toc437261049)

[Annex E Example Messages 26](#_Toc437261050)

[Annex F Example Request with Attachment 28](#_Toc437261051)

[Annex G Example Request with MTOM Attachment 29](#_Toc437261052)

# License

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/.

# Introduction

This specification describes the X-Road message protocol version 4.0. This protocol is used between information systems and security servers in the X-Road system. The protocol is implemented as a profile of the SOAP 1.1 protocol [SOAP]. Because this protocol inherits the general model, the transport mechanism, and the error handling mechanism of the base SOAP protocol, these issues are not discussed separately in this specification.

Chapters 2 and 3, as well as Annex A, Annex B of this specification contain normative information. All the other chapters are informative in nature. All the references are normative.

This specification does not include option for partially implementing the protocol – the conformant implementation must implement the entire specification.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document (in uppercase, as shown) are to be interpreted as described in [RFC2119].

## Terms and Abbreviations

* **X-Road member** – natural or legal person who uses functionality offered by X-Road
* **Subsystem** – represents part of an information system belonging to an X-Road member. X-Road member can divide its information system to subsystems, although this is not mandatory
* **Central service** – centrally defined alias to a concrete service implemented by an X-Road member or a subsystem.
* **X-Road service** – SOAP-based web service that is offered by an X-Road member or by a subsystem and that can be used by other X-Road members or subsystems.

## References

[SOAP] Simple Object Access Protocol (SOAP) 1.1, 2000.

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels, Internet  
 Engineering Task Force, 1997.

[DSIG] XML Signature Syntax and Processing Version 2.0, 2013.

[SOAPATT] SOAP Messages with Attachments, 2000.

[WSDL] Web Services Description Language (WSDL) 1.1, 2001.

[XSD1] XML Schema Part 1: Structures Second Edition, 2004.

[XSD2] XML Schema Part 2: Datatypes Second Edition, 2004.

[MTOM] SOAP 1.1 Binding for MTOM 1.0, 2006

[SWAREF] Attachments Profile Version 1.0, 2004

## Identifying Entities

Significant entities in the X-Road system have globally unique identifiers. Identifiers consist of an object type and a sequence of hierarchical codes.

All the identifiers start with the code identifying the instance of the X-Road system. Typically, this should be the ISO code of the country running the X-Road instance, optionally amended with a suffix corresponding to the environment. For example, for Estonia, the production environment is designated as “EE”, whereas the test environment is “EE-test”. The codes for X-Road instances are the only ones that need to be globally unique. All other parts of the identifiers are managed by X-Road instances.

Next, we will describe how globally unique identifiers are constructed for various types of entities. When representing entities as strings the format *T:C1/C2/...* is used, where *T* is type of the entity and *C1, C2, ...* are the component codes. Note: the given format is only used in this document. In messages and configuration files, the identifiers are represented in XML format described in Section 2.1.

* **X-Road member** – *MEMBER:[X-Road instance]/[member class]/[member code]*. The identifier consists of the following components:

– code corresponding to the X-Road instance;

– code identifying the member class (e.g., government agency, private enterprise, physical person. Typically, member codes are issued by an authority guaranteeing the uniqueness of the codes within the given member class); and

– member code that uniquely identifies the given X-Road member within its member class.

Example: identifier MEMBER:EE/BUSINESS/123456789 represents an organization registered in Estonia (EE) with a business registry code (BUSINESS) of 123456789.

* **Subsystem** – *SUBSYSTEM:[subsystem owner]/[subsystem code]*. Identifier for a subsystem consists of the identifier of the X-Road member that owns the subsystem, and a subsystem code. The subsystem code is chosen by the X-Road member and it must be unique among the subsystems of this member.  
    
  Example: SUBSYSTEM:EE/BUSINESS/123456789/highsecurity identifies a subsystem with code highsecurity belonging to the X-Road member from the previous example (MEMBER:EE/BUSINESS/123456789).
* **Service** – *SERVICE:[service provider]/[service code]/[service version]*. Identifier for a service consists of an identifier of the service provider (either an X-Road member or a subsystem), service code, and version. The service code is chosen by the service provider. Version is optional and can be used to distinguish between technically incompatible versions of the same basic service.  
    
  Example: SERVICE:EE/BUSINESS/123456789/highsecurity/getSecureData/v1 identifies version v1 of service getSecureData that is offered by subsystem SUBSYSTEM:EE/BUSINESS/123456789/highsecurity.
* **Central service** – *CENTRALSERVICE:/[X-Road instance]/[service code]*. The list of central services is managed by the X-Road governing agency who also assigns unique codes for these services.  
    
  Example: CENTRALSERVICE:EE/populationRegister\_personData identifies a central service that returns person data from the national Population Register.

# Format of Messages

The messages in this protocol are based on SOAP 1.1 format [SOAP].

## Identifiers

This section describes XML-based data formats for expressing the identifiers described informally in Section 1.3. The data structures and elements defined in this section will be located under namespace http://x-road.eu/xsd/identifiers. The complete XML Schema is shown in Annex A .

The following listing shows the header of the schema definition.

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified"

targetNamespace="http://x-road.eu/xsd/identifiers"

xmlns="http://x-road.eu/xsd/identifiers">

The XRoadIdentifierType complex type serves as the base for all other identifier types (derived by restriction). It contains a union of all fields that can be present in different identifiers. The attribute objectType contains the type of the identifier and can be used, for example, to distinguish between X-Road member and subsystem identifiers without resorting to conditions that check for presence of individual fields.

<xs:complexType name="XRoadIdentifierType">

<xs:sequence>

<xs:element minOccurs="0" ref="xRoadInstance"/>

<xs:element minOccurs="0" ref="memberClass"/>

<xs:element minOccurs="0" ref="memberCode"/>

<xs:element minOccurs="0" ref="subsystemCode"/>

<xs:element minOccurs="0" ref="serviceCode"/>

<xs:element minOccurs="0" ref="serviceVersion"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required"/>

</xs:complexType>

The enumeration XRoadObjectType lists all possible values of the objectType attribute.

<xs:simpleType name="XRoadObjectType">

<xs:restriction base="xs:string">

<xs:enumeration value="MEMBER"/>

<xs:enumeration value="SUBSYSTEM"/>

<xs:enumeration value="SERVICE"/>

<xs:enumeration value="CENTRALSERVICE"/>

</xs:restriction>

</xs:simpleType>

Next, we define elements and attributes used in the XRoadIdentifierType.

<xs:element name="xRoadInstance" type="xs:string"/>

<xs:element name="memberClass" type="xs:string"/>

<xs:element name="memberCode" type="xs:string"/>

<xs:element name="subsystemCode" type="xs:string"/>

<xs:element name="serviceCode" type="xs:string"/>

<xs:element name="serviceVersion" type="xs:string"/>

<xs:attribute name="objectType" type="XroadObjectType"/>

Finally, we define complex types for representing concrete types of identifiers. First, the XRoadClientIdentifierType is used to represent identifiers that can be used by the service clients, namely X-Road members and subsystems.

<xs:complexType name="XRoadClientIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="memberClass"/>

<xs:element ref="memberCode"/>

<xs:element minOccurs="0" ref="subsystemCode"/>

</xs:sequence>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

The XRoadServiceIdentifierType can be used to represent identifiers of services.

<xs:complexType name="XRoadServiceIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="memberClass"/>

<xs:element ref="memberCode"/>

<xs:element minOccurs="0" ref="subsystemCode"/>

<xs:element ref="serviceCode"/>

<xs:element minOccurs="0" ref="serviceVersion"/>

</xs:sequence>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

The XRoadCentralServiceIdentifierType can be used to represent identifiers of central services.

<xs:complexType name="XRoadCentralServiceIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="serviceCode"/>

</xs:sequence>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

## Message Headers

This section describes additional SOAP headers that are used by the X-Road system. It makes use of data types specified in Section 2.1. The header fields are described in Table 1.

Table 1. Supported header fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Mandatory/Optional** | **Description** |
| client | XRoadClientIdentifierType | M | Identifies a service client – an entity that initiates the service call. |
| service | XRoadServiceIdentifierType | O | Identifies the service that is invoked by the request. |
| centralService | XRoadCentralServiceIdentifierType | O | Identifies the central service that is invoked by the request. |
| id | string | M | Unique identifier for this message. The recommended form of message ID is UUID. |
| userId | string | O | User whose action initiated the request. The user ID should be prefixed with two-letter ISO country code (e.g., EE12345678901). |
| issue | string | O | Identifies received application, issue or document that was the cause of the service request. This field may be used by the client information system to connect service requests (and responses) to working procedures. |
| protocolVersion | string | M | X-Road message protocol version. The value of this field MUST be 4.0 |
| requestHash | string | O | For responses, this field contains a Base64 encoded hash of the request SOAP message. This field is automatically filled in by the service provider's security server. |
| requestHash/ @algorithmId | string | M | Identifies the hash algorithm that was used to calculate the value of the requestHash field. The algorithms are specified as URIs listed in the XML-DSIG specification [DSIG]. |

When a service client sends a request to the security server, exactly one of the fields service or centralService MUST be present. If the centralService field is used, the security server resolves the central service and automatically fills in the service field with the identifier of the concrete service that implements the central service. Thus, in the request sent to the service, both fields MAY be present (the service field is always present).

When responding, the service MUST copy all the header fields from the request to the response in the exact same sequence with the exact same values. The XML namespace prefix of the header fields has no significance to the security server, but the prefix must reference the same namespace as in the request.

The requestHash field is used to create a strong connection between a request and a response. Thus, it is possible to prove, for example, that a certain registry record is returned in response to a certain query. The requestHash is computed from the byte contents of the SOAP request message using the algorithm from the requestHash/@algorithmId field. The byte contents of the SOAP request message are:

* in case the request has no attachments – the byte contents of the HTTP POST request sent to the service client's security server;
* in case the request is a multipart MIME message with attachments – the byte contents of the first part of the multipart message. Messages with attachments are described in more detail in Section 2.4.

The requestHash field MUST be automatically created by the service provider's security server when receiving the service response message and MUST be verified by the service client's security server.

The request message SHOULD NOT contain the requestHash field. The response message sent by service to the service provider's security server SHOULD NOT contain the requestHash field. If the response message contains the requestHash field, the service provider's security server MUST ignore the field and replace it with the created field.

The requestHash field SHOULD NOT be described in the service WSDL.

Content-type HTTP header of the client request message is preserved in the security servers and is forwarded to the service. All other HTTP headers of the client request message are not preserved by the security servers and are not forwarded to the service.

Content-type HTTP header of the service response message is preserved in the security servers and is forwarded to the client. All other HTTP headers of the service response are not preserved by security servers and are not forwarded to the client.

Starting with X-Road message protocol version 4.0 any protocols with the same major version number are compatible. Minor versions are used to describe backwards compatible changes, such as addition of optional headers.

## Message Body

The message body MUST use Document/Literal-Wrapped SOAP encoding convention. According to this convention, both the body of the request and the response must be wrapped in an element. The element names of the request and response are correlated – if the request element is named foo then the response element is named fooResponse. Additionally, the name of the wrapper element of the request must match the serviceCode element of the service header field.

## Attachments

In case the message has attachments, it MUST be formatted as a multipart MIME message, with the SOAP request and its attachments being separate parts of the message. The SOAP request must be the first part. The resulting MIME message MUST be structured in accordance with the specification for SOAP messages with attachments [SOAPATT] and the request SOAP part's *Content-Transfer-Encoding* MIME header value MUST be “8bit”. MIME headers of each part of the message are preserved without modification in the security server. For an example request that contains attachments see Annex F.

Additionally, MTOM-encoded [MTOM] messages are supported in the security server – the security server accepts MIME multipart messages where the content-type of the SOAP part is “application/xop+xml”.

## Fault Messages

For technical errors the security server must return a SOAP Fault message[[1]](#footnote-1). The SOAP Fault message contains the information about the error, such as error code, error message etc. The SOAP Fault MAY contain X-Road Headers and it MAY be described in the service WSDL.

# Describing Services

## General

Services are described using the Web Services Description Language (WSDL) 1.1 [WSDL].

X-Road supports versioned services. Different versions of the service represent minor technical changes in the service description. For example, a new version must be created when restructuring the service description (e.g., renaming or refactoring types in the XML Schema) or when changing types or names of fields. However, when the service semantics or data content of messages changes, a new service with a new code must be created.

In the context of service provision contracts, services are considered without version, meaning that all versions of the same service are considered to be equivalent. This also applies to access control restrictions applied in security servers – i.e., access control restrictions are specified for a service code without version. In order for this to work, all versions of the same service must implement the same contract.

## Describing Services with WSDL

Service descriptions are written in the WSDL language, subject to the following restrictions and extensions.

The combination of WSDL binding style/use MUST be document/literal (binding *style=”document”; use=”literal”*). The WSDL must conform to the following rules[[2]](#footnote-2):

1. *“****Only 'One' Part Definition in the Input & Output Messages in WSDL*** *'Wrapped' is a form of document/literal. When defining a WS-I compliant document/literal service, there can be at most one body part in your input message and at most one body part in your output message. You do \*not\* define each method parameter as a separate part in the message definition. (The parameters are defined in the WSDL 'types' section, instead).”*
2. ***“'Part' Definitions are wrapper elements*** *Each part definition must reference an element (not a type, type is used in RPC) defined to make it document style of messaging. This element definition can be imported, or included in the types section of the WSDL document. These element definitions are 'wrapper' elements (hence the name of this convention). Define the input and output parameters as element structures within these wrapper elements."*
3. *“****Child Elements of 'Part' Element Type will be SEI Method parameter*** *An input wrapper element must be defined as a complex type that is a sequence of elements. Each child element-type in that sequence will be generated (while using code generation tool on WSDL) as a parameter of the operation in the service interface.”*
4. *“****Input Wrapper Element name should match with Operation name*** *The name of the input wrapper element must be the same as the web service operation name in WSDL.”*
5. *“****<Output Wrapper Element Name> = <Operation Name> + 'Response'*** *The name of the output wrapper element could be (but doesn't have to be) the operation name appended with 'Response' (e.g., if the operation name is 'add', the output wrapper element should be called 'addResponse').”*
6. *“****In the WSDL Binding section, soap:binding style = 'document'*** *Since, the style is document/literal for this wrapped pattern, hence in the binding definition, the soap:binding should specify style='document' (although this is the default value, so the attribute may be omitted), and the soap:body definitions must specify use='literal' and nothing else. You must not specify the namespace or encodingStyle attributes in the soap:body definition.”*

The input and output parameters of the services are described using XML Schema 1.1 [XSD1, XSD2].

In order to avoid confusion from the client's side in determining whether an empty response indicates a silent error or simply contains no output records, it is a good practice to design the output of a data-returning service in such a way that for any service calls the response contains at least one non-empty scalar parameter. This parameter can be a non-technical error message (technical error messages should be returned with SOAP Fault messages). For error message examples see Annex D.

For a service description WSDL example and messages conforming to this description see Annex C and Annex E, respectively.

The traditional way of describing SOAP attachments in WSDL documents [WSDL] is considered to be legacy approach because it cannot bind SOAP envelope with attachments. Instead of that it is recommended to use swaRef types [SWAREF]. It is also possible to describe attachments using MTOM [MTOM].

For example of swaRef and MTOM on-the-wire messages with attachments see Annex F and Annex G respectively. For both swaRef and MTOM service description WSDL examples see Annex C.

Table 2 lists elements that can be added to a WSDL description to transfer information specific to X-Road. The namespace prefix xrd is bound to namespace "http://x-road.eu/xsd/xroad.xsd".

Table 2. WSDL elements for X-Road services

|  |  |
| --- | --- |
| **Field** | **Description** |
| /definitions/binding/operation/@name | Code of the service |
| /definitions/binding/operation/xrd:version | Version of the service |
| /definitions/portType/operation/documentation/xrd:title | Title of the service (for displaying to users) |
| /definitions/portType/operation/documentation/xrd:notes | Description of the service (for displaying to users) |
| /definitions/portType/operation/documentation/xrd:techNotes | Description of the service (for developers) |

# XML Schema for Identifiers

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema elementFormDefault="qualified" jxb:version="2.1" targetNamespace="http://x-road.eu/xsd/identifiers" xmlns="http://x-road.eu/xsd/identifiers" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:complexType name="XRoadIdentifierType">

<xs:annotation>

<xs:documentation>Globally unique identifier in the XROAD system. Identifier consists of object type specifier and list of hierarchical codes (starting with code that identifiers the XROAD instance).</xs:documentation>

</xs:annotation>

<xs:sequence>

<xs:element minOccurs="0" ref="xRoadInstance"/>

<xs:element minOccurs="0" ref="memberClass"/>

<xs:element minOccurs="0" ref="memberCode"/>

<xs:element minOccurs="0" ref="subsystemCode"/>

<xs:element minOccurs="0" ref="groupCode"/>

<xs:element minOccurs="0" ref="serviceCode"/>

<xs:element minOccurs="0" ref="serviceVersion"/>

<xs:element minOccurs="0" ref="securityCategoryCode"/>

<xs:element minOccurs="0" ref="serverCode"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required"/>

</xs:complexType>

<xs:simpleType name="XRoadObjectType">

<xs:annotation>

<xs:documentation>Enumeration for XROAD identifier types.</xs:documentation>

</xs:annotation>

<xs:restriction base="xs:string">

<xs:enumeration value="MEMBER"/>

<xs:enumeration value="SUBSYSTEM"/>

<xs:enumeration value="SERVER"/>

<xs:enumeration value="GLOBALGROUP"/>

<xs:enumeration value="LOCALGROUP"/>

<xs:enumeration value="SECURITYCATEGORY"/>

<xs:enumeration value="SERVICE"/>

<xs:enumeration value="CENTRALSERVICE"/>

</xs:restriction>

</xs:simpleType>

<xs:element name="xRoadInstance" type="xs:string">

<xs:annotation>

<xs:documentation>Identifies the XROAD instance. This field is applicable to all identifier types.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="memberClass" type="xs:string">

<xs:annotation>

<xs:documentation>Type of the member (company, government institution, private person, etc.)</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="memberCode" type="xs:string">

<xs:annotation>

<xs:documentation>Code that uniquely identifies a member of given member type.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="subsystemCode" type="xs:string">

<xs:annotation>

<xs:documentation>Code that uniquely identifies a subsystem of given XROAD member.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="groupCode" type="xs:string">

<xs:annotation>

<xs:documentation>Code that uniquely identifies a global group in given XROAD instance.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="serviceCode" type="xs:string">

<xs:annotation>

<xs:documentation>Code that uniquely identifies a service offered by given XROAD member or subsystem.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="serviceVersion" type="xs:string">

<xs:annotation>

<xs:documentation>Version of the service.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="securityCategoryCode" type="xs:string">

<xs:annotation>

<xs:documentation>Code that uniquely identifies security category in a given XROAD instance.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="serverCode" type="xs:string">

<xs:annotation>

<xs:documentation>Code that uniquely identifies security server offered by a given XROAD member or subsystem.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:attribute name="objectType" type="XRoadObjectType"/>

<xs:complexType name="XRoadClientIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="memberClass"/>

<xs:element ref="memberCode"/>

<xs:element minOccurs="0" ref="subsystemCode"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required"/>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="XRoadServiceIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="memberClass"/>

<xs:element ref="memberCode"/>

<xs:element minOccurs="0" ref="subsystemCode"/>

<xs:element ref="serviceCode"/>

<xs:element minOccurs="0" ref="serviceVersion"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required" fixed="SERVICE"/>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="XRoadSecurityCategoryIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="securityCategoryCode"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required" fixed="SECURITYCATEGORY"/>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="XRoadCentralServiceIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="serviceCode"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required" fixed="CENTRALSERVICE"/>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="XRoadSecurityServerIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="memberClass"/>

<xs:element ref="memberCode"/>

<xs:element ref="serverCode"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required" fixed="SERVER"/>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="XRoadGlobalGroupIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="xRoadInstance"/>

<xs:element ref="groupCode"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required" fixed="GLOBALGROUP"/>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="XRoadLocalGroupIdentifierType">

<xs:complexContent>

<xs:restriction base="XRoadIdentifierType">

<xs:sequence>

<xs:element ref="groupCode"/>

</xs:sequence>

<xs:attribute ref="objectType" use="required" fixed="LOCALGROUP"/>

</xs:restriction>

</xs:complexContent>

</xs:complexType>

</xs:schema>

# XML Schema for Messages

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema elementFormDefault="qualified"

targetNamespace="http://x-road.eu/xsd/xroad.xsd"

xmlns="http://x-road.eu/xsd/xroad.xsd"

xmlns:id="http://x-road.eu/xsd/identifiers"

xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:import namespace="http://www.w3.org/XML/1998/namespace"

schemaLocation="http://www.w3.org/2009/01/xml.xsd"/>

<xs:import id="id" namespace="http://x-road.eu/xsd/identifiers"

schemaLocation="http://x-road.eu/xsd/identifiers.xsd"/>

<!-- Header elements -->

<xs:element name="client" type="id:XRoadClientIdentifierType">

<xs:annotation>

<xs:documentation>Identies service client</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="service" type="id:XRoadServiceIdentifierType">

<xs:annotation>

<xs:documentation>Identies the service

that is invoked by the request</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="centralService"

type="id:XRoadCentralServiceIdentifierType">

<xs:annotation>

<xs:documentation>Identies the central service

that is invoked by the request.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="id" type="xs:string">

<xs:annotation>

<xs:documentation>Unique identier

for this message</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="userId" type="xs:string">

<xs:annotation>

<xs:documentation>User whose action initiated

the request</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="requestHash">

<xs:annotation>

<xs:documentation>Base64 encoded hash of

the SOAP request message</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute name="algorithmId" type="xs:string">

<xs:annotation>

<xs:documentation>Identies hash algorithm

that was used to calculate the value

of the requestHash field.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name="issue" type="xs:string">

<xs:annotation>

<xs:documentation>Identies received application, issue or document

that was the cause of the service request.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="protocolVersion" type="xs:string">

<xs:annotation>

<xs:documentation>X-Road message protocol version</xs:documentation>

</xs:annotation>

</xs:element>

<!-- Elements describing other elements and operations-->

<xs:element name="version" type="xs:string">

<xs:annotation>

<xs:documentation>Version of the service</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="title">

<xs:annotation>

<xs:documentation>Title of the service</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute default="en" ref="xml:lang"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name="notes">

<xs:annotation>

<xs:documentation>Notes for user</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute ref="xml:lang" default="en" />

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name="techNotes">

<xs:annotation>

<xs:documentation>Notes for technical stuff</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute ref="xml:lang" default="en" />

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

</xs:schema>

# Example WSDL

<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions targetNamespace="http://producer.x-road.eu"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

xmlns:tns="http://producer.x-road.eu"

xmlns:xrd="http://x-road.eu/xsd/xroad.xsd"

xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"

xmlns:xmime="http://www.w3.org/2005/05/xmlmime"

xmlns:ref="http://ws-i.org/profiles/basic/1.1/xsd"

xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<wsdl:types>

<xs:schema targetNamespace="http://producer.x-road.eu"

xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:import namespace="http://x-road.eu/xsd/xroad.xsd"

schemaLocation="http://x-road.eu/xsd/xroad.xsd" />

<xs:import namespace="http://ws-i.org/profiles/basic/1.1/xsd"

schemaLocation="http://ws-i.org/profiles/basic/1.1/swaref.xsd" />

<xs:import namespace="http://www.w3.org/2005/05/xmlmime"

schemaLocation="http://www.w3.org/2005/05/xmlmime" />

<xs:complexType name="fault">

<xs:sequence>

<xs:element name="faultCode" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Fault Code</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

<xs:element name="faultString" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Fault explanation</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

<xs:element name="exampleService">

<xs:complexType>

<xs:sequence>

<xs:element name="exampleInput" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example input</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="exampleServiceResponse">

<xs:complexType>

<xs:sequence>

<xs:element name="exampleOutput" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example output</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

<xs:element name="fault" type="tns:fault"

minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="exampleServiceSwaRef">

<xs:complexType>

<xs:sequence>

<xs:element name="exampleInput" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example input</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

<xs:element name="exampleAttachment" type="ref:swaRef">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example Attachment (with swaRef

description)</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="exampleServiceSwaRefResponse">

<xs:complexType>

<xs:sequence>

<xs:element name="exampleOutput" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example output</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

<xs:element name="fault" type="tns:fault"

minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="exampleServiceMtom">

<xs:complexType>

<xs:sequence>

<xs:element name="exampleInput" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example input</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

<xs:element name="exampleAttachment"

type="xs:base64Binary"

xmime:expectedContentTypes="application/octet-stream">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example MTOM

Attachment</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="exampleServiceMtomResponse">

<xs:complexType>

<xs:sequence>

<xs:element name="exampleOutput" type="xs:string">

<xs:annotation>

<xs:appinfo>

<xrd:title>Example output</xrd:title>

</xs:appinfo>

</xs:annotation>

</xs:element>

<xs:element name="fault" type="tns:fault"

minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

</wsdl:types>

<wsdl:message name="exampleService">

<wsdl:part name="exampleService" element="tns:exampleService" />

</wsdl:message>

<wsdl:message name="exampleServiceResponse">

<wsdl:part name="exampleServiceResponse"

element="tns:exampleServiceResponse" />

</wsdl:message>

<wsdl:message name="exampleServiceSwaRef">

<wsdl:part name="exampleServiceSwaRef"

element="tns:exampleServiceSwaRef" />

</wsdl:message>

<wsdl:message name="exampleServiceSwaRefResponse">

<wsdl:part name="exampleServiceSwaRefResponse"

element="tns:exampleServiceSwaRefResponse" />

</wsdl:message>

<wsdl:message name="exampleServiceMtom">

<wsdl:part name="exampleServiceMtom"

element="tns:exampleServiceMtom" />

</wsdl:message>

<wsdl:message name="exampleServiceMtomResponse">

<wsdl:part name="exampleServiceMtomResponse"

element="tns:exampleServiceMtomResponse" />

</wsdl:message>

<wsdl:message name="requestHeader">

<wsdl:part name="client" element="xrd:client" />

<wsdl:part name="service" element="xrd:service" />

<wsdl:part name="id" element="xrd:id" />

<wsdl:part name="userId" element="xrd:userId" />

<wsdl:part name="issue" element="xrd:issue" />

<wsdl:part name="protocolVersion" element="xrd:protocolVersion" />

</wsdl:message>

<wsdl:portType name="exampleServicePort">

<wsdl:operation name="exampleService">

<wsdl:documentation>

<xrd:title>Title of exampleService</xrd:title>

<xrd:notes>Technical notes for exampleService:

This is a simple SOAP service.</xrd:notes>

</wsdl:documentation>

<wsdl:input name="exampleService" message="tns:exampleService" />

<wsdl:output name="exampleServiceResponse"

message="tns:exampleServiceResponse" />

</wsdl:operation>

<wsdl:operation name="exampleServiceSwaRef">

<wsdl:documentation>

<xrd:title>Title of exampleServiceSwaRef</xrd:title>

<xrd:notes>Technical notes for exampleServiceSwaRef:

This is a SOAP service with

swaRef attachment.</xrd:notes>

</wsdl:documentation>

<wsdl:input name="exampleServiceSwaRef"

message="tns:exampleServiceSwaRef" />

<wsdl:output name="exampleServiceSwaRefResponse"

message="tns:exampleServiceSwaRefResponse" />

</wsdl:operation>

<wsdl:operation name="exampleServiceMtom">

<wsdl:documentation>

<xrd:title>Title of exampleServiceMtom</xrd:title>

<xrd:notes>Technical notes for exampleServiceMtom:

This is a SOAP service with

MTOM attachment.</xrd:notes>

</wsdl:documentation>

<wsdl:input name="exampleServiceMtom"

message="tns:exampleServiceMtom" />

<wsdl:output name="exampleServiceMtomResponse"

message="tns:exampleServiceMtomResponse" />

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="exampleServicePortSoap11"

type="tns:exampleServicePort">

<soap:binding style="document"

transport="http://schemas.xmlsoap.org/soap/http" />

<wsdl:operation name="exampleService">

<soap:operation soapAction="" style="document" />

<xrd:version>v1</xrd:version>

<wsdl:input name="exampleService">

<soap:body use="literal" />

<soap:header message="tns:requestHeader"

part="client" use="literal" />

<soap:header message="tns:requestHeader"

part="service" use="literal" />

<soap:header message="tns:requestHeader"

part="id" use="literal" />

<soap:header message="tns:requestHeader"

part="userId" use="literal" />

<soap:header message="tns:requestHeader"

part="issue" use="literal" />

<soap:header message="tns:requestHeader"

part="protocolVersion" use="literal"/>

</wsdl:input>

<wsdl:output name="exampleServiceResponse">

<soap:body use="literal" />

<soap:header message="tns:requestHeader"

part="client" use="literal" />

<soap:header message="tns:requestHeader"

part="service" use="literal" />

<soap:header message="tns:requestHeader"

part="id" use="literal" />

<soap:header message="tns:requestHeader"

part="userId" use="literal" />

<soap:header message="tns:requestHeader"

part="issue" use="literal" />

<soap:header message="tns:requestHeader"

part="protocolVersion" use="literal" />

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="exampleServiceSwaRef">

<soap:operation soapAction="" style="document" />

<xrd:version>v1</xrd:version>

<wsdl:input>

<!-- MIME description is required according to WS-I Attachments

Profile Version 1.0: R2902 A SENDER MUST NOT send a

message using SOAP with Attachments if the corresponding

wsdl:input or wsdl:output element in the wsdl:binding does

not specify the WSDL MIME Binding.

The WSDL 1.1 specification does not specify whether the

soap:header element is permitted as a child of the

mime:part element along with the soap:body element. But

WS-I Attachments Profile Version 1.0 recommends including

both soap:header and soap:body as a content of mime:part.

However it should be noted that some tools like for

example SoapUI and Eclipse Web Services Explorer assume

that soap:header elements are children of wsdl:input or

wsdl:output elements. -->

<mime:multipartRelated>

<mime:part>

<soap:body use="literal" />

<soap:header message="tns:requestHeader"

part="client" use="literal" />

<soap:header message="tns:requestHeader"

part="service" use="literal" />

<soap:header message="tns:requestHeader"

part="id" use="literal" />

<soap:header message="tns:requestHeader"

part="userId" use="literal" />

<soap:header message="tns:requestHeader"

part="issue" use="literal" />

<soap:header message="tns:requestHeader"

part="protocolVersion" use="literal" />

</mime:part>

</mime:multipartRelated>

</wsdl:input>

<wsdl:output>

<soap:body use="literal" />

<soap:header message="tns:requestHeader"

part="client" use="literal" />

<soap:header message="tns:requestHeader"

part="service" use="literal" />

<soap:header message="tns:requestHeader"

part="id" use="literal" />

<soap:header message="tns:requestHeader"

part="userId" use="literal" />

<soap:header message="tns:requestHeader"

part="issue" use="literal" />

<soap:header message="tns:requestHeader"

part="protocolVersion" use="literal" />

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="exampleServiceMtom">

<soap:operation soapAction="" style="document" />

<xrd:version>v1</xrd:version>

<wsdl:input>

<!-- MTOM does not require MIME description -->

<soap:header message="tns:requestHeader"

part="client" use="literal" />

<soap:header message="tns:requestHeader"

part="service" use="literal" />

<soap:header message="tns:requestHeader"

part="id" use="literal" />

<soap:header message="tns:requestHeader"

part="userId" use="literal" />

<soap:header message="tns:requestHeader"

part="issue" use="literal" />

<soap:header message="tns:requestHeader"

part="protocolVersion" use="literal" />

<soap:body use="literal" />

</wsdl:input>

<wsdl:output>

<soap:body use="literal"/>

<soap:header message="tns:requestHeader"

part="client" use="literal" />

<soap:header message="tns:requestHeader"

part="service" use="literal" />

<soap:header message="tns:requestHeader"

part="id" use="literal" />

<soap:header message="tns:requestHeader"

part="userId" use="literal" />

<soap:header message="tns:requestHeader"

part="issue" use="literal" />

<soap:header message="tns:requestHeader"

part="protocolVersion" use="literal" />

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="producerPortService">

<wsdl:port name="exampleServicePortSoap11"

binding="tns:exampleServicePortSoap11">

<soap:address location="http://foo.bar.baz" />

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

# Example Fault Messages

This section contains example SOAP Fault messages.

D.1 Technical

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

<SOAP-ENV:Body>

<SOAP-ENV:Fault>

<faultcode>Server.ClientProxy.ServiceFailed.MissingBody</faultcode>

<faultstring>Malformed SOAP message: body missing</faultstring>

<faultactor></faultactor>

<detail>

<faultDetail xmlns="">f31e7451-f0ac-48f6-9f05-1f0459e48eea</faultDetail>

</detail>

</SOAP-ENV:Fault>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

D.2 Non-technical

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:ns1="http://producer.x-road.eu"

xmlns:id="http://x-road.eu/xsd/identifiers"

xmlns:xrd="http://x-road.eu/xsd/xroad.xsd">

<SOAP-ENV:Header>

<xrd:client id:objectType="MEMBER">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER1</id:memberCode>

<id:subsystemCode>SUBSYSTEM1</id:subsystemCode>

</xrd:client>

<xrd:service id:objectType="SERVICE">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER2</id:memberCode>

<id:subsystemCode>SUBSYSTEM2</id:subsystemCode>

<id:serviceCode>test</id:serviceCode>

<id:serviceVersion>v1</id:serviceVersion>

</xrd:service>

<xrd:id>4894e35d-bf0f-44a6-867a-8e51f1daa7e0</xrd:id>

<xrd:userId>EE12345678901</xrd:userId>

<xrd:issue>12345</xrd:issue>

<xrd:protocolVersion>4.0</xrd:protocolVersion>

<xrd:requestHash

algorithmId="http://www.w3.org/2001/04/xmlenc#sha512">

8r+UeXoU2WiEXRMdES8KBLhdQV/lt1DA+rLi2EUC239k

OvBWGcBjYde27YIZtNQObsyHFQfX0V6pQ6LH3KS1Hw==

</xrd:requestHash>

</SOAP-ENV:Header>

<SOAP-ENV:Body>

<ns1:exampleServiceResponse>

<exampleOutput />

<fault>

<faultCode>test\_failed</faultCode>

<faultString>Could not read test parameters</faultString>

</fault>

</ns1:exampleServiceResponse >

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

# Example Messages

This section contains example request and example response messages for a random number generator service.

E.1 Request

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:ns1="http://producer.x-road.eu"

xmlns:xrd="http://x-road.eu/xsd/xroad.xsd"

xmlns:id="http://x-road.eu/xsd/identifiers">

<SOAP-ENV:Header>

<xrd:client id:objectType="MEMBER">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER1</id:memberCode>

<id:subsystemCode>SUBSYSTEM1</id:subsystemCode>

</xrd:client>

<xrd:service id:objectType="SERVICE">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER2</id:memberCode>

<id:subsystemCode>SUBSYSTEM2</id:subsystemCode>

<id:serviceCode>exampleService</id:serviceCode>

<id:serviceVersion>v1</id:serviceVersion>

</xrd:service>

<xrd:id>4894e35d-bf0f-44a6-867a-8e51f1daa7e0</xrd:id>

<xrd:userId>EE12345678901</xrd:userId>

<xrd:issue>12345</xrd:issue>

<xrd:protocolVersion>4.0</xrd:protocolVersion>

</SOAP-ENV:Header>

<SOAP-ENV:Body>

<ns1:exampleService>

<exampleInput>foo</exampleInput>

</ns1:exampleService>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

E.2 Response

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:ns1="http://producer.x-road.eu"

xmlns:id="http://x-road.eu/xsd/identifiers"

xmlns:xrd="http://x-road.eu/xsd/xroad.xsd">

<SOAP-ENV:Header>

<xrd:client id:objectType="MEMBER">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER1</id:memberCode>

<id:subsystemCode>SUBSYSTEM1</id:subsystemCode>

</xrd:client>

<xrd:service id:objectType="SERVICE">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER2</id:memberCode>

<id:subsystemCode>SUBSYSTEM2</id:subsystemCode>

<id:serviceCode>exampleService</id:serviceCode>

<id:serviceVersion>v1</id:serviceVersion>

</xrd:service>

<xrd:id>4894e35d-bf0f-44a6-867a-8e51f1daa7e0</xrd:id>

<xrd:userId>EE12345678901</xrd:userId>

<xrd:issue>12345</xrd:issue>

<xrd:protocolVersion>4.0</xrd:protocolVersion>

<xrd:requestHash

algorithmId="http://www.w3.org/2001/04/xmlenc#sha512">

29KTVbZf83XlfdYrsxjaSYMGoxvktnTUBTtA4BmSrh1e

gtRtvR9VY8QycYaVdsKtGJIh/8CpucYWPbWfaIgJDQ==

</xrd:requestHash>

</SOAP-ENV:Header>

<SOAP-ENV:Body>

<ns1:exampleServiceResponse>

<exampleOutput>bar</exampleOutput>

</ns1:exampleServiceResponse>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

# Example Request with Attachment

.. other transport headers ...

Content-Type: multipart/related; type="text/xml"; start="<rootpart>"; boundary="MIME\_boundary"

MIME-Version: 1.0

--MIME\_boundary

Content-Type: text/xml; charset=UTF-8

Content-Transfer-Encoding: 8bit

Content-ID: <rootpart>

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:ns1="http://producer.x-road.eu"

xmlns:xrd="http://x-road.eu/xsd/xroad.xsd"

xmlns:id="http://x-road.eu/xsd/identifiers">

<SOAP-ENV:Header>

<xrd:client id:objectType="MEMBER">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER1</id:memberCode>

<id:subsystemCode>SUBSYSTEM1</id:subsystemCode>

</xrd:client>

<xrd:service id:objectType="SERVICE">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER2</id:memberCode>

<id:subsystemCode>SUBSYSTEM2</id:subsystemCode>

<id:serviceCode>exampleService</id:serviceCode>

<id:serviceVersion>v1</id:serviceVersion>

</xrd:service>

<xrd:id>4894e35d-bf0f-44a6-867a-8e51f1daa7e0</xrd:id>

<xrd:userId>EE12345678901</xrd:userId>

<xrd:issue>12345</xrd:issue>

<xrd:protocolVersion>4.0</xrd:protocolVersion>

</SOAP-ENV:Header>

<SOAP-ENV:Body>

<ns1:exampleServiceSwaRef>

<exampleInput>foo</exampleInput>

<exampleAttachment>cid:data.bin</exampleAttachment>

</ns1:exampleServiceSwaRef>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

--MIME\_boundary

Content-Type: application/octet-stream; name=data.bin

Content-Transfer-Encoding: base64

Content-ID: <data.bin>

Content-Disposition: attachment; name="data.bin"; filename="data.bin"

VGhpcyBpcyBhdHRhY2htZW50Lg0K

--MIME\_boundary--

# Example Request with MTOM Attachment

... other transport headers ...

Content-Type: multipart/related; type="application/xop+xml"; start="<rootpart>"; start-info="text/xml"; boundary="MIME\_boundary"

MIME-Version: 1.0

--MIME\_boundary

Content-Type: application/xop+xml; charset=UTF-8; type="text/xml"

Content-Transfer-Encoding: 8bit

Content-ID: <rootpart>

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:ns1="http://producer.x-road.eu"

xmlns:xrd="http://x-road.eu/xsd/xroad.xsd"

xmlns:id="http://x-road.eu/xsd/identifiers">

<SOAP-ENV:Header>

<xrd:client id:objectType="MEMBER">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER1</id:memberCode>

<id:subsystemCode>SUBSYSTEM1</id:subsystemCode>

</xrd:client>

<xrd:service id:objectType="SERVICE">

<id:xRoadInstance>EE</id:xRoadInstance>

<id:memberClass>GOV</id:memberClass>

<id:memberCode>MEMBER2</id:memberCode>

<id:subsystemCode>SUBSYSTEM2</id:subsystemCode>

<id:serviceCode>exampleService</id:serviceCode>

<id:serviceVersion>v1</id:serviceVersion>

</xrd:service>

<xrd:id>4894e35d-bf0f-44a6-867a-8e51f1daa7e0</xrd:id>

<xrd:userId>EE12345678901</xrd:userId>

<xrd:issue>12345</xrd:issue>

<xrd:protocolVersion>4.0</xrd:protocolVersion>

</SOAP-ENV:Header>

<SOAP-ENV:Body>

<ns1:exampleServiceMtom>

<exampleInput>foo</exampleInput>

<exampleAttachment>

<inc:Include href="cid:data.bin"

xmlns:inc="http://www.w3.org/2004/08/xop/include" />

</exampleAttachment>

</ns1:exampleServiceMtom>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

--MIME\_boundary

Content-Type: application/octet-stream; name=data.bin

Content-Transfer-Encoding: base64

Content-ID: <data.bin>

Content-Disposition: attachment; name="data.bin"; filename="data.bin"

VGhpcyBpcyBhdHRhY2htZW50Lg0K

--MIME\_boundary--

1. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#\_Toc478383507 [↑](#footnote-ref-1)
2. http://www.ibm.com/developerworks/library/ws-usagewsdl/ [↑](#footnote-ref-2)