

X-Road Service protocol

# **X-Road Protocol for Data Exchange Between Service Requester and Service Provider**

---

## **Requirements for Information Systems and Adapter Servers**

**Specification**

Version 5.0

42 pp

# Version History

<b>Date</b>	<b>Version</b>	<b>Description</b>	<b>Author</b>
28 October 2010	5.0	First version	

## **ANNOTATION**

This document describes the X-Road protocol for data exchange between service requesters and service providers. The document includes message examples.

## CONTENTS

<b>1 INTRODUCTION</b> .....	<b>6</b>
1.1 TERMS AND DEFINITIONS .....	6
1.2 PURPOSE OF THE DOCUMENT AND INTENDED AUDIENCE .....	6
1.3 OPERATION OF X-ROAD FROM DEVELOPER'S VIEWPOINT .....	6
1.4 TECHNICAL ASPECTS OF CONNECTING TO X-ROAD .....	7
<b>2 DEFINITIONS AND ABBREVIATIONS</b> .....	<b>8</b>
2.1 STANDARDS USED .....	8
2.1.1 SOAP .....	8
2.1.2 WSDL .....	8
2.1.3 Namespaces .....	8
2.1.4 Services .....	8
<b>3 STRUCTURE OF MESSAGES IN DATA EXCHANGE PROTOCOL</b> .....	<b>9</b>
3.1 DIVISION OF DATA BETWEEN COMPONENTS .....	9
3.2 ELEMENTS OF HEADER .....	9
3.3 MESSAGE PRESENTATION IN SOAP .....	10
3.3.1 Restrictions .....	10
3.3.2 Service request .....	10
3.3.3 Service response .....	11
3.3.4 Asynchronous messaging .....	11
3.4 ERROR MESSAGES .....	11
3.4.1 Standard SOAP error message .....	12
3.4.2 Non-technical SOAP error message .....	12
<b>4 FORMAT OF DATA SERVICES</b> .....	<b>13</b>
4.1 NAMES OF SERVICES AND PARAMETERS .....	13
4.2 WSDL FORMAT .....	13
4.3 PUBLICATION OF SERVICE DESCRIPTIONS .....	15
4.4 MESSAGES WITH MIME ATTACHMENTS .....	16
<b>5 METASERVICES</b> .....	<b>18</b>
5.1 OVERVIEW .....	18
5.2 DESCRIPTION OF METASERVICES .....	18
5.2.1 Service for listing databases (listProducers) .....	18
5.2.2 Service for listing allowed methods (allowedMethods) .....	19
5.2.3 Service for querying the next unsent asynchronous message (asyncNext) .....	19
5.2.4 Service for querying the last successfully sent asynchronous message (asyncLast) .....	19
5.2.5 Service for listing adapter server's methods (listMethods) .....	20

5.2.6 Service for monitoring an adapter server (testSystem).....	20
5.2.7 Service for monitoring the service provider (getState).....	21
5.2.8 Service for determining service costs (getCharge) .....	21
5.2.9 Service for loading a classifier (loadClassification) .....	21
5.2.10 Service for entering a legacy system (legacyXYZ) .....	22
<b>6 REFERENCES .....</b>	<b>24</b>
<b>7 EXAMPLES .....</b>	<b>25</b>
7.1 EXAMPLES OF SERVICES .....	25
7.1.1 Service listProducers .....	25
7.1.2 Service allowedMethods .....	25
7.1.3 Service listMethods .....	26
7.1.4 Service testSystem .....	27
7.1.5 Service getState .....	27
7.1.6 Service loadClassification .....	28
7.1.7 Service legacyXYZ .....	30
7.1.8 Data service .....	31
7.1.9 Service with a MIME-attachment .....	32
7.1.10 Standard error with header .....	34
7.2 SAMPLE SERVICE DESCRIPTIONS (WSDL).....	35
7.2.1 Sample description of database services .....	35

# 1 INTRODUCTION

## 1.1 TERMS AND DEFINITIONS

X-Road	National Data Security Layer
HTTP	Hypertext Transfer Protocol
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language

## 1.2 PURPOSE OF THE DOCUMENT AND INTENDED AUDIENCE

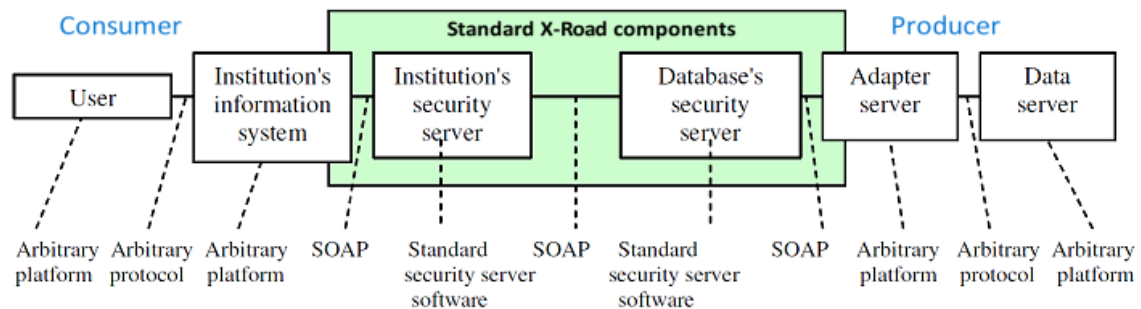
The purpose of this document is describing the X-Road protocol on a level sufficient for connecting service providers and service requesters to X-Road. Connecting to X-Road means either making your services available via X-Road, or using services provided by X-Road in your information system. The document explains what restrictions X-Road imposes on the standard protocols used, what types of services can be used/implemented via X-Road, and how services should be described.

The reader is expected to be familiar with the SOAP protocol. The purpose of this document is not to describe the SOAP protocol, as there are plenty of other sources of information. Besides, X-Road is not tied to any particular implementation of those protocols. The document is intended for developers of information systems (service requesters) and service providers who will implement X-Road connections.

## 1.3 OPERATION OF X-ROAD FROM DEVELOPER'S VIEWPOINT

The data exchange protocol of X-Road regulates data exchange between an information system (service requester) and the adapter server of a service provider (typically it is a database). The information system of an institution provides service to end users, utilizing platforms and protocols independent of X-Road. The X-Road related communication of both the information system and the adapter server occurs only via corresponding security servers. The security server of an institution makes a connection to the security server of the provider and forwards the service invocation (request or query) received from the information system.

The adapter server of the service provider modifies queries arriving from X-Road to a form that can be processed by the service provider's data server (which is independent of X-Road), and returns the data server's response in a form suitable for X-Road.



The SOAP protocol can be used in the communication between the institution's information system and the security server, as well as between the provider's adapter server and the security server. The infosystem's ability to access the service via X-Road doesn't depend on the protocols used by either security server

The X-Road infrastructure contains many other components, besides the components described above. Those other components will not be described in this document because they are not necessary for implementing X-Road services. The service requesters and service providers that connect to X-Road will only communicate with their own security server on X-Road.

## 1.4 TECHNICAL ASPECTS OF CONNECTING TO X-ROAD

Adapter servers and information systems can be implemented by any software vendor. The only technical requirements presented by X-Road are:

- the software has to communicate with the security server via the standardized X-Road protocol;
- the institutions's information system must have sufficient security level to be connected to X-Road.

Implementing an adapter server means creating a SOAP server and publishing the necessary data services, as well as metaservices obligatory for the adapter servers. The parameters of the adapter server are stored in the service provider's security server which will access the adapter server.

To integrate an institution's information system with X-Road, a SOAP client has to be created for forwarding service invocations. The information system will then pass all service invocation requests via http or https to an URL located in the institution's security server: `/cgi-bin/consumer_proxy`. Standard libraries are usually used for creating SOAP servers and clients. A set of SOAP libraries can be found at <http://www.soapware.org/directory/4/implementations>.

## 2 DEFINITIONS AND ABBREVIATIONS

### 2.1 STANDARDS USED

#### 2.1.1 SOAP

SOAP [SOAP] is an XML-based data exchange protocol. Information systems and adapter servers can communicate with security servers via protocol that conforms to the SOAP specification, with restrictions that will be described in this document.

#### 2.1.2 WSDL

WSDL [WSDL] is a language for describing Web services. X-Road service descriptions conform to the WSDL specification, with restrictions that will be described in this document.

#### 2.1.3 Namespaces

The document uses the following namespace prefixes.

- **xrd** - refers to namespace <http://x-rd.net/xsd/xroad.xsd>
- **xsd** - refers to namespace <http://www.w3.org/2001/XMLSchema>
- **xsi** - refers to namespace <http://www.w3.org/2001/XMLSchema-instance>
- **SOAP-ENV** - refers to namespace <http://schemas.xmlsoap.org/soap/envelope/>
- **SOAP-ENC** - refers to namespace <http://schemas.xmlsoap.org/soap/encoding/>

#### 2.1.4 Services

The SOAP methods executed on X-Road form X-Road services. An X-Road service is a two-stage data exchange between a service requester and a service provider, initiated by the service requester information system by sending a query (a message with service input), which is then processed by the service provider to return query results (a message with service output). From a technical viewpoint, it is irrelevant for X-Road whether the aim of the service is to use the stored data in a database to generate a report, to store the received data in the database, or something else.

There are two types of services: **data services** and **metaservices**. Data services are services specific to a particular service provider and usually created individually for that service provider. To give access to these services is the main goal of X-Road. The input and output of data services of a service provider are specified in the provider's documentation. Data services belong to the namespace of their corresponding provider: <http://name.CC.x-rd.net/producer/>, where *CC* is a two-letter country code and *name* is the name of a service provider.

Metaservices are auxiliary services for obtaining information necessary to perform data services. The input, output and semantics of metaservices are standardized and will be described in this document. They are similar in all servers providing metaservices. Metaservices belong to X-Road namespace <http://x-rd.net/xsd/xroad.xsd>.

## 3 STRUCTURE OF MESSAGES IN DATA EXCHANGE PROTOCOL

### 3.1 DIVISION OF DATA BETWEEN COMPONENTS

Data exchanged through X-Road is structured in three parts: header, query (*<request>*), and result (*<response>*). A component usually contains sub-elements, but can also contain scalar values.

Messages that request a service contain a header and the *<request>* component.

Messages that are responses to service requests contain a header, the *<request>* component and the *<result>* component. In such case, the query and response headers are identical. Also, the *<request>* component is also the same for both query and response.

The header component is presented inside the SOAP envelope header. The query and body components are presented as elements *<request>* and *<response>* immediately below the root element of the SOAP envelope body.

All X-Road messages are encoded as UTF-8.

### 3.2 ELEMENTS OF HEADER

The header consists of the following elements, which must be child tags of the *<SOAP-ENV:Header>* tag:

Tag	Data type	Description
consumer	string	Name of the service requester institution
producer	string	Name of the service provider
userId	string	ID code of the person invoking the service This code could be in one of the two forms: <i>CCX</i> or <i>CC:R:X</i> where <ul style="list-style-type: none"> <li>• <i>CC</i> - two-letter country code in capital letters, e.g. EE for Estonia</li> <li>• <i>R</i> - ID of the registry</li> <li>• <i>X</i> - registry value or code</li> </ul> The first form is used in countries where exists unique and universal person identification code. For example: EE12345678901 or UK:XYZ:123456789
id	string	Unique identifier
service	string	Name of the service to be invoked
issue	string	Name of file or document related to the service invocation

The *id* field is a unique identifier, which can contain both numbers and Latin letters ([a-zA-Z]) and hyphen ('-'). It is generated by information systems, that must ensure that the identifier is globally unique, for example, by combining a random number with the name of the institution.

The service name in the header must match the name of the method that is executed. It is duplicated to facilitate processing of signed queries.

In addition to the elements listed above, the header of a query can also contain other header elements described in the X-Road namespace:

Tag	Data type	Description
unit	string	Registration code of the institution or its unit on whose behalf the service is used
position	string	Organizational position or role of the person invoking the service
userName	string	Name of the person invoking the service
async	boolean	Specifies asynchronous service. If the value is "true", then the security server performs the service call asynchronously.
authenticator	string	Authentication method, one of the following: <ul style="list-style-type: none"> <li>• <b>ID-CARD</b> – use a certificate of identity</li> <li>• <b>CERT</b> – use another certificate</li> <li>• <b>EXTERNAL</b> – authenticate through a third-party service</li> <li>• <b>PASSWORD</b> – authenticate with a password</li> </ul> Details of the authenticator (e.g. the identification of a bank for external authentication) can be given in brackets after the authentication method.
paid	string	The amount of money (in the smallest monetary unit, e.g cents) paid for invoking the service
currency	string	three-letter currency code in capital letters

Required elements of the query header are listed in the WSDL file.

### 3.3 MESSAGE PRESENTATION IN SOAP

**Note:** In this section and the sections that follow, the elements that contain values specific to the service are printed in **boldface**.

#### 3.3.1 Restrictions

SOAP is presented using the *Document/Literal Wrapped* style [WSDL-STYLE].

In X-Road message headers and metaservices, all elements must be presented in the single-reference form. Polymorphic elements cannot be used, that is, all parameters in a message must be accessible via only one path that defines the parameter uniquely.

#### 3.3.2 Service request

The service request has the following structure.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    Header components
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:service xmlns:m="URI">
      <request>
        Request components
      </request>
    </m:service>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

By **header** and **query components**, it is meant the SOAP representation of the component's all sub-parameters (if the component is a structured parameter) or the component's value (if the component is a scalar parameter). Elements can have other attributes besides those listed here.

The element `<request>` can be omitted, if the service does not contain any parameters.

### 3.3.3 Service response

The service response has the following structure.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    Header components
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:serviceResponse xmlns:m="URI">
      <request>
        Request components
      </request>
      <response>
        Response components
      </response>
    </m:serviceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

By **header**, **request**, and **response components**, it is meant the SOAP representation of the component's all sub-parameters (if the component is a structured parameter) or the component's value (if the component is a scalar parameter). Elements can have other attributes besides those listed here.

Elements can have other attributes besides those listed here.

The element `<request>` can be omitted, if the service does not contain any parameters or if it is a response to a metaservice.

### 3.3.4 Asynchronous messaging

A message is considered asynchronous if in the header the field "async" is set to "true". When the message is sent to the database, the response is given by its security server. On success, the server returns an empty response (i.e., an acknowledgement or a receipt) without elements (`<response/>`).

## 3.4 ERROR MESSAGES

The output of a service can be an error message. Error messages can be presented in two ways, depending on the contents of the message.

- To notify about service failure unrelated to the user, use a standard SOAP error message.
- To report an input error (for example, the user entered incorrect data or the query returned no results), the element `<response>` should contain a structure that presents the error string in the elements `faultCode` and `faultString`.

### 3.4.1 Standard SOAP error message

If possible, a SOAP error message should also contain a header, but it is not obligatory.

Error codes correspond to codes in the SOAP specification. Codes belonging to class *Client* refer to errors in the service input composed by the information system. Codes belonging to class *Server* refer to errors not related to service input. Error codes generated by security servers are listed in the document [SERR].

The parameter *faultactor* is used to store the identity of the component that created the error message, if possible.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    contents of header
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>fault code</faultcode>
      <faultactor>fault actor</faultactor>
      <faultstring>fault string</faultstring>
      <detail>fault details</detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Under the header component is given a SOAP representation of all sub-parameters of the corresponding component.

### 3.4.2 Non-technical SOAP error message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    contents of header
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:serviceResponse xmlns:m="URI">
      <request>
        contents of the request component
      </request>
      <response>
        <faultCode>fault code</faultCode>
        <faultString>fault string</faultString>
      </response>
    </serviceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

By header, request, and response, it is meant the SOAP representation of the component's all sub-parameters (if the component is a structured parameter) or the component's value (if the component is a scalar parameter).

## 4 FORMAT OF DATA SERVICES

### 4.1 NAMES OF SERVICES AND PARAMETERS

A service is identified by its name. The name of a service consists of three components separated by dots, in the form *producer.method.version* (for example, *land-cadastre.query1.v1*), where:

- *producer* is the name of the service provider. The name is fixed upon its joining X-Road;
- *method* is short name of the service, unique among this provider's services;
- *version* is version of the service in the form vN, where N is version number.

All three components must conform to the character set allowed in DNS (i.e. Latin letters, numbers, and hyphen). The short name of a data service cannot match the name of any metaservice.

Any individual version of a service is fixed and must not change over time. Any change in the description of a service requires creation of either a new version of the service, or, if the semantics of the service changed, creation of a new service. The semantics of a service must be the same across all versions of the service.

Names of the service parameters can consist only of Latin letters, numbers, underscores, and dots. The name of a parameter cannot begin with an underscore or a number.

### 4.2 WSDL FORMAT

The WSDL format of service description conforms to the WSDL specification, with the restrictions listed below.

- The combination *binding style/use* must be *document/literal* (that is, *binding style="document"; use="literal"*).
- Query and response parameters are described as an XML schema [XSD].

Also, the following elements have been added to WSDL to store information necessary for X-Road.

Element location (XPath)	Value/Description
/definitions/service/port/xrd:address	No value, but the element has the attribute <i>producer</i> , the value of which is the name of the service provider.
/definitions/service/port/xrd:title	Title of the service provider (for displaying to users)
/definitions/binding/operation/xrd:version	Service version
/definitions/binding/operation/xrd:charge	Payment options (if money is charged for the use)

Element location (XPath)	Value/Description
	of the service)
/definitions/binding/operation/xrd:charge/xrd:account	Bank account of the payee
/definitions/binding/operation/xrd:charge/xrd:receiverName	Name of the payee
/definitions/binding/operation/xrd:charge/xrd:message	Description of payment
/definitions/binding/operation/xrd:charge/xrd:amount	Amount payable. The element can have the attribute <i>chargeType</i> , the value of which is the type of payee to whom the sum applies.
/definitions/binding/operation/xrd:noContent	Handling of parameter elements with no content. <i>null</i> – no content means that the corresponding parameter has no value; <i>empty</i> – no content means that the corresponding parameter is null (this is the default)
/definitions/binding/operation/xrd:requireContent	Requirements for values in mandatory fields: <i>true</i> – mandatory fields must contain values; <i>false</i> – mandatory field can have null value. A field is mandatory if the value of the attribute <i>minOccurs</i> > 0.
/definitions/portType/operation/documentation/xrd:title	Service title (for displaying to users)
/definitions/portType/operation/documentation/xrd:notes	Service comments (for displaying to users)
/definitions/portType/operation/documentation/xrd:techNotes	Service comments (for displaying to developers)
/definitions/portType/operation/documentation/xrd:actionTitle	Title for the submit action (for example, <i>Save</i> )
//annotation/appinfo/xrd:title	Parameter title (for displaying to users)
//annotation/appinfo/xrd:notes	Parameter comments (for displaying to users)
//annotation/appinfo/xrd:techNotes	Parameter comments (for displaying to developers)
//annotation/appinfo/xrd:fieldType	Parameter type, one of the following: <i>textarea</i> – corresponds to the HTML <code>&lt;textarea&gt;</code> <code>&lt;/textarea&gt;</code> element; <i>comment</i> – text displayed to the user (the value is a constant set in the WSDL or in a complex query). <b>Note:</b> For the service provider to be able to assume that the user has not changed the value of a parameter, the attribute <i>default</i> is required.
//annotation/appinfo/xrd:fieldRows	Recommended number of rows on screen for the input field (the <i>rows</i> attribute in the HTML element <code>&lt;textarea rows="..." cols="..."&gt;</code> <code>&lt;/textarea&gt;</code> )
//annotation/appinfo/xrd:fieldCols	Recommended number of columns on screen for the input field (the <i>cols</i> attribute in the HTML element <code>&lt;textarea rows="..." cols="..."&gt;</code> <code>&lt;/textarea&gt;</code> )
//annotation/appinfo/xrd:fieldSize	The length of an input field on the screen (the <i>size</i> attribute in the HTML element <code>&lt;input type="text" size="..."&gt;</code> )
//annotation/appinfo/xrd:wildcard	List of metasymbols allowed
//annotation/appinfo/xrd:ref	Name of the parameter to which the current parameter is related.

If the element `<xtee:address>` is present, then the service is considered accessible via the X-Road protocol.

In describing X-Road services, the following elements are obligatory: name and title of the service provider; name, heading, and version number of the service. It is recommended to specify a name for each parameter.

In the elements `<xrd:title>`, `<xrd:notes>` and `<xrd:techNotes>`, the attribute `xml:lang` can be used to store the language in which the value of the element is presented. If the attribute is missing, the default value "en" (English) is presumed.

When invoking services, metasymbols can be used in parameter values if the metasymbols are listed in the description of the service. In service description, all metasymbols allowed in parameters should be listed in the element `<xrd:wildcard>`. The following symbols can be used.

*	(asterisk)	Can be used to replace any number of arbitrary symbols
?	(question mark)	Can be used to replace one arbitrary symbol
-	(hyphen)	Can be used to denote an interval
P	(letter P)	Value is considered a prefix
S	(letter S)	Value is considered a substring

To avoid the possibility that an error causes the absence of service output and another error causes the absence of the corresponding error message, it is recommended to design the output of a data service so that the body of the query contains at least one non-empty scalar parameter, independent of how the service was invoked. This parameter could be a non-technical error message.

A default or fixed value of a scalar parameter is described as a value of the *default* or *fixed* attribute, respectively.

In addition to describing the default or fixed value as a constant, it may also be given by a reference to a special variable defined in the information system. In that case, the information system uses the value of the variable as a default or fixed value. A default or fixed value is referencing a variable if the value of the attribute starts with the prefix string 'xrdvar:'. The rest of the value is considered to be the name of the variable, which may contain only Latin characters ([a-zA-Z]), numbers, and underscores.

Service requests with no output description (for which the WSDL element `/binding/operation/output` is missing) will be forwarded asynchronously.

### 4.3 PUBLICATION OF SERVICE DESCRIPTIONS

Service provider administrator publishes descriptions of the provider's data services as a WSDL-format file in the provider's adapter server.

The description of services can be distributed across multiple WSDL files. The URL of the WSDL file is stored in the service provider's security server. If necessary, the file may link to other files either on public Internet or in the same folder as the main WSDL file, provided that the location is accessible to the security server.

If the WSDL files are on a Web server accessible to the institution's information system, then the service descriptions can be downloaded with ordinary HTTP GET queries. If the Web server is not accessible, then the service descriptions can be downloaded via the institutions's security server over the URL: `http://secureproxy/cgi-bin/uriproxy?uri=URI`, where:

- *secureproxy* – security server's address
- *URI* – address of the WSDL or schema

Similarly, it is possible to download a WSDL file that describes the services by accessing the security server through the URL: `http://secureproxy/cgi-bin/uriproxy?producer=name`, where:

- *secureproxy* – security server's address,
- *name* – name of the service provider.

In addition to WSDL service descriptions, the adapter server can provide a description of services in the form of an XForms file. To access the file, access the security server through the URL: `http://secureproxy/cgi-bin/uriproxy?service=name`, where:

- *secureproxy* – security server's address
- *name* – name of the service in form *producer.method.version*.

#### 4.4 MESSAGES WITH MIME ATTACHMENTS

Messages with SOAP protocol are allowed to contain MIME attachments, if the wire format of the message corresponds to specification [SA] and service description corresponds to specification [WSDL] and following constraints are in use::

- HTTP header field *Content-Type* has value of *multipart/related*;
- MIME container header field *Content-Type* contains parameter *type* with value of *text/xml*;
- MIME container header field *Content-Type* contains parameter *boundary* with value of the boundary string used to separate parts of the MIME message;
- the first part of a MIME container is always a SOAP envelope, which contains the header and `<request>`;
- the *Content-Transfer-Encoding* of the part containing the SOAP envelope is *8bit*;
- the SOAP envelope contains references to all attachments in the message, by using the value of *Content-Id* in the attachment header.
- the message will be encoded as MIME container if WSDL describes MIME *binding* for the method;
- if the message is encoded as MIME container then values of all scalar elements of the input with type of either *xsd:base64Binary* or *xsd:hexBinary* will be sent as attachments;
- if the service query contains links to MIME attachments, then the response message contains SHA-512 hash values of the attached data.

Attachments are the preferred way of transporting huge data, because transmitting large attachments requires less security server resources than transmitting the same data within a SOAP message without attachments.

## 5 METASERVICES

### 5.1 OVERVIEW

An institutions's information system can access the following metaservices:

- listProducers
- allowedMethods
- asyncNext
- asyncLast
- getState

An institutions's information system may also have access to some of the following metaservices on some databases:

- getCharge
- loadClassification
- legacyXYZ (where XYZ is an arbitrary text characteristic to that query)

An adapter server is required to implement the following metaservices:

- listMethods

An adapter server may be required to implement the following metaservices, depending on agreements:

- testSystem

An adapter server may implement the following metaservices:

- getCharge
- loadClassification
- legacyXYZ

### 5.2 DESCRIPTION OF METASERVICES

#### 5.2.1 Service for listing databases (listProducers)

The service *xrd.listProducers* is implemented in the institution's security server and returns a list of existing databases. The service allows automatic retrieval of service provider names. With such a list, it is later possible to retrieve a list of all available services through the *allowedMethods* metaservice.

Method call:

- Header - Not used

- <request> – Not used
- <response> – Contains the following sequence of *item* elements:

Tag	Data type	Description
name	string	Short name of the X-Road service provider
description	string	Full name of the service provider

### 5.2.2 Service for listing allowed methods (allowedMethods)

The service *producer.allowedMethods* (where *producer* is the name of a service provider) is implemented in the service provider's security server. The service is called by the institution's information system and returns list of methods available to the institution.

Method call:

- Header – Required
- <request> – Not used
- <response> – Contains the following sequence of elements:

Tag	Data type	Description
item	string	Name of service in the form <i>producer.method.version</i>

### 5.2.3 Service for querying the next unsent asynchronous message (asyncNext)

The service *xrd.asyncNext* is implemented in the institution's security server. The service returns the ID of the oldest message in the queue of asynchronous messages.

Method call:

- Header – Not used
- <request> (in query) – Name of the service provider or an empty string
- <request> (in response) – Not used
- <response> – The ID of the oldest unsent message in the queue

According to the query parameter, asynchronous messages are chosen. If a name is specified in the request, the specified provider's que will be inspected. If the name is empty, the queue with the oldest first message (that is, one that arrived at the security server first) will be inspected.

Response contains a string (possibly empty) that is the Id of the first message in the queue being inspected. If the queue was not found or it is empty, the response will be an empty string. There is a different queue for each database, because security server is allowed to send messages to different databases independently from each other as long as the order of messages within a database is maintained.

### 5.2.4 Service for querying the last successfully sent asynchronous message (asyncLast)

Service *xrd.asyncLast* is implemented in the security server. The service returns the nonce (*id*) of the last successfully sent asynchronous message.

Method call:

- Header – Not used
- <request> (query) – Name of the service provider or an empty string
- <request> (response) – Not used
- <response> – The ID of the last successfully sent asynchronous message in the queue

If input string is not empty then it is the name of the database whose queue should be inspected. Otherwise the last successful asynchronous message will be reported no matter which database it was sent to.

Response contains a string (possibly empty) that is Id of the last successfully sent message in the queue being inspected. If no asynchronous message has been sent successfully then the response will be an empty string.

### 5.2.5 Service for listing adapter server's methods (listMethods)

Service *system.listMethods* is implemented in the adapter server. The service returns a list of all methods implemented in the adapter server. This service is meant to be called only by the database's own security server.

Method call:

- Header – Not used
- <request> – Not used
- <response> – Contains the following sequence of elements:

Tag	Data type	Description
item	string	name of query in the form <i>producer.method.version</i>

### 5.2.6 Service for monitoring an adapter server (testSystem)

Service *system.testSystem* is implemented in the adapter server. The service is used to verify that the adapter server and the service provider are operating correctly. The service can be used only in the local security server of the provider.

Method call:

- Header – Not used
- <request> – Not used
- <response> – Not used

If the adapter server fails, it should generate an error message (see chapter "Standard SOAP error message").

### 5.2.7 Service for monitoring the service provider (getState)

The service *producer.getState* (where *producer* is the name of a service provider) is implemented in the provider's security server. The service returns the status of the service provider that the security server has received from the adapter server after a *testSystem* call.

Method call:

- Header – Required
- <request> – Not used
- <response> – State of the provider (*int* type)

The <response> component can have one of the following values: 0 – state unknown; 1 – status OK; 2 – error.

### 5.2.8 Service for determining service costs (getCharge)

The service *producer.getCharge* (where *producer* is the name of a service provider) can be implemented in the adapter server. The service allows the service provider to assign a cost, which may change over time, to the invocation of the service. The service returns the fee/charge for invoking a particular service (identified by name) for the user whose code is listed in the query header.

Method call:

- Header – Required
- <request> – Name of the service in the form *producer.query.version*
- <response> – Contains the following elements:

Tag	Data type	Description
amount	int	Amount of fee
currency	string	Currency code

### 5.2.9 Service for loading a classifier (loadClassification)

Service *producer.loadClassification* (where *producer* is the name of a service provider) returns one of the following: a list of classifiers, the contents of a classifier, or a subset of the contents of a classifier.

Method call:

- Header – requierd
- <request> – empty string
- <response> – Element *classificationNames* containing the following sequence of elements:

Tag	Data type	Description
item	string	Name of the classifier

The contents of a classifier:

- Header – required
- <request> – The following sequence of elements:

Tag	Data type	Description
name	string	Name of classifier
subset	string	A subset of the classifier's contents
from	date	Include data from the given start date
max	string	Limit the number of records to be returned

- <response> – the classifier structure

If the <request> component is an empty string, then a list of classifier names is returned.

A subset of a classifier is defined by the date parameter *from*, which sets the earliest date for modifications to the classification specified, for example, using an LDAP identifier.

The response structure corresponds to the LDAP tree structure of response data. A LDAP record is presented as a structure with sub-parameters containing fields of that record and lists of sub-records. Elements of lists in subrecords are anonymous; names of lists are values of the *objectClass* field of the LDAP records corresponding to elements. If a field of a LDAP record has several values, then the field is presented as a list of anonymous values, and the name of the list is the name of the field.

### 5.2.10 Service for entering a legacy system (legacyXYZ)

Services with names in the form *producer.legacyXYZ* (where *producer* is the name of a service provider and *XYZ* is arbitrary text characteristic to that service) are services for logging in to legacy systems. This type of service allows to use the authentication and authorization methods of an information system integrated with X-Road in other information systems.

Method call:

- Header – Required
- <request> – Contains the following sequence of elements:

Tag	Data type	Description
item	string	Name of the service allowed to the person, in the form <i>producer.query.version</i>

- <response> – URL of the service

The information system, when creating the query, adds all names of legacy system services allowed for that person at the given service provider to the body of the query.

The security server of the provider checks, whether the institution is allowed to invoke all the services listed. If the list contains services that the institution has no right to invoke, then the security server will respond with a fault message and will refuse to invoke the service.

The URL in the service response is a unique URL created by adapter server and/or legacy system's server that uniquely identifies the session, for example:

<https://legacy.example.com/cgi-bin/legacy.cgi?id=6d82fbea82752264a9724919cabf9e11>

The URL in the service response should be available over HTTPS (not HTTP). The service must be implemented so that the *id* is unique and not possible to guess. The information system servicing the URL must guarantee the expiration of the URL.

## 6 REFERENCES

- [WSDL] Web Services Description Language (WSDL) 1.1  
<http://www.w3.org/TR/wsdl>
- [WSDL-STYLE] Which style of WSDL should I use?  
<http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/#N1021A>
- [XSD] XML Schema Part 2: Datatypes  
<http://www.w3.org/TR/xmlschema-2/>
- [SOAP] Simple Object Access Protocol (SOAP) 1.1  
<http://www.w3.org/TR/SOAP/>
- [SA] SOAP Messages with Attachments  
<http://www.w3.org/TR/SOAP-attachments>
- [SERR] SOAP error codes used in X-Road  
soap\_errors.rtf

## 7 EXAMPLES

### 7.1 EXAMPLES OF SERVICES

#### 7.1.1 Service listProducers

##### Query:

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <xrd:listProducers xmlns:xrd="http://x-rd.net/xsd/xroad.xsd"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

##### Response:

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
  <xrd:listProducersResponse xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
    <response>
      <item>
        <name>car-registry</name>
        <description>National Traffic Register</description>
      </item>
      <item>
        <name>pr</name>
        <description>Population Registry</description>
      </item>
      <item>
        <name>land-cadastre</name>
        <description>Land Cadastre</description>
      </item>
    </response>
  </xrd:listProducersResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

#### 7.1.2 Service allowedMethods

##### Query:

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>land-cadastre</xrd:producer>
```

```

    <xrd:userId>EE:PIN:abc4567</xrd:userId>
    <xrd:id >411d6755661409fed365ad8135f8210be07613da</xrd:id>
    <xrd:service>land-cadastre.allowedMethods</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:allowedMethods/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Response:**

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>land-cadastre</xrd:producer>
    <xrd:userId>EE:PIN:abc4567</xrd:userId>
    <xrd:id>411d6755661409fed365ad8135f8210be07613da</xrd:id>
    <xrd:service>land-cadastre.allowedMethods</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:allowedMethodsResponse>
      <response>
        <item>land-cadastre.cu.v1</item>
        <item>land-cadastre.cuAddres.v1</item>
      </response>
    </xrd:allowedMethodsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**7.1.3 Service listMethods****Query:**

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <xrd:listMethods xmlns:xrd="http://x-rd.net/xsd/xroad.xsd"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Response:**

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <xrd:listMethodsResponse xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
      <response>
        <item>land-cadastre.cu.v1</item>
        <item>land-cadastre.cuAddres.v1</item>
        <item>land-cadastre.legacy1.v1</item>
      </response>
    </xrd:listMethodsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

    </xrd:listMethodsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

### 7.1.4 Service testSystem

#### Query:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <xrd:testSystem xmlns:xrd="http://x-rd.net/xsd/xroad.xsd"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

#### Response:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <xrd:testSystemResponse xmlns:xrd="http://x-rd.net/xsd/xroad.xsd"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

### 7.1.5 Service getState

#### Query:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>land-cadastre</xrd:producer>
    <xrd:userId>EE:PIN:abc4567</xrd:userId>
    <xrd:id >411d6755661409fed365ad8135f8210be07613da</xrd:id>
    <xrd:service>land-cadastre.getState</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:getState/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

#### Response:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>land-cadastre</xrd:producer>

```

```

    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id >411d6755661409fed365ad8135f8210be07613da</xrd:id>
    <xrd:service>land-cadastre.getState</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:getStateResponse/>
      <response>1</response>
    <xrd:getStateResponse/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

### 7.1.6 Service loadClassification

#### Query that returns a list of classifiers:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>assert</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>3e47be25da43528deb639f5965d58b4d21bbc173</xrd:id>
    <xrd:service>assert.loadClassification</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:loadClassification>
      <request/>
    </xrd:loadClassification>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

#### Response:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>assert</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>3e47be25da43528deb639f5965d58b4d21bbc173</xrd:id>
    <xrd:service>assert.loadClassification</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:loadClassificationResponse>
      <request/>
      <response>
        <classificationNames>
          <item>EHAK</item>
          <item>abc</item>

```

```

    </classificationNames>
  </response>
</xrd:loadClassificationResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

### Query that loads a classifier:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>assert</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>00f340cc5da4d3102c0859cc5f2ae2547b07b2c3</xrd:id>
    <xrd:service>assert.loadClassification</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:loadClassification>
      <request>
        <name>EHAK</name>
        <subset/>
        <from/>
        <max/>
      </request>
    </xrd:loadClassification>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

### Response:

```

<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>assert</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>00f340cc5da4d3102c0859cc5f2ae2547b07b2c3</xrd:id>
    <xrd:service>assert.loadClassification</xrd:service>
    <xrd:issue />
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xrd:loadClassificationResponse>
      <request>
        <name>EHAK</name>
        <subset/>
        <from/>
        <max/>
      </request>
      <response>
        <classifications>

```

```

<item>
  <l>37</l>
  <objectclass>xrdLocality</objectclass>
  <cn>Harju</cn>
  <ehakType>0</ehakType>
  <description>province</description>
  <subclassifications>
    <item>
      <l>112</l>
      <objectclass>xrdLocality</objectclass>
      <cn>Aegviidu</cn>
      <ehakType>1</ehakType>
      <description>vald</description>
    </item>
  </subclassifications>
</item>
<item>
  <l>39</l>
  <objectclass>xrdLocality</objectclass>
  <cn>Hiiu</cn>
  <ehakType>0</ehakType>
  <description>province</description>
</item>
</classifications>
<classificationNames/>
</response>
</xrd:loadClassificationResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

### 7.1.7 Service legacyXYZ

#### Query:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>assert</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>3d0a5efdb4093dc04a8be68cd3227f009bd5c35d</xrd:id>
    <xrd:service>assert.legacy1.v1</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns4:legacy1 xmlns:ns4="http://assert.ee.x-rd.net/producer/">
      <request>
        <item>assert.aspirant</item>
        <item>assert.houses</item>
        <item>assert.legacyQuery1</item>
        <item>assert.legacyQuery2</item>
        <item>assert.business</item>
      </request>
    </ns4:legacy1>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Response:**

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>assert</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>3d0a5efdb4093dc04a8be68cd3227f009bd5c35d</xrd:id>
    <xrd:service>assert.legacy1.v1</xrd:service>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns4:legacy1Response xmlns:ns4="http://assert.ee.x-rd.net/producer/">
      <request>
        <item>assert.aspirant</item>
        <item>assert.houses</item>
        <item>assert.legacyQuery1</item>
        <item>assert.legacyQuery2</item>
        <item>assert.business</item>
      </request>
      <response>
        <url>
          https://legacy.assert.ee:4000/legacyid=6d82fbea82752264a9724919cabf9e11
        </url>
      </response>
    </ns4:legacy1Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**7.1.8 Data service****Query:**

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>land-cadastre</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>3aed1ae3813eb7fbed9396fda70ca1215d3f3fe1</xrd:id>
    <xrd:service>land-cadastre.cuAddres.v1</xrd:service>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns4:cuAddres
      xmlns:ns4="http://land-cadastre.ee.x-rd.net/producer/">
      <request>
        <province>0037</province>
        <localGovernment>0784</localGovernment>
        <cuOfficialName>cross road square*</cuOfficialName>
        <cuMax>100</cuMax>
      </request>
    </ns4:cuAddres>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

</ns4:cuAdres>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## Response:

```

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
    <xrd:consumer>10239452</xrd:consumer>
    <xrd:producer>land-cadastre</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>3aed1ae3813eb7fbed9396fda70ca1215d3f3fe1</xrd:id>
    <xrd:service>land-cadastre.cuAdres.v1</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns4:cuAdresResponse
      xmlns:ns4="http://land-cadastre.ee.x-rd.net/producer/">
      <request>
        <province>0037</province>
        <localGovernment>0784</localGovernment>
        <cuOfficialName>cross road square*</cuOfficialName>
        <cuMax>100</cuMax>
      </request>
      <response>
        <item>
          <cadastralUnit>012:345:67</cadastralUnit>
          <cuLocation>Cross Rode Square 1A</cuLocation>
          <province>0037</province>
          <localGovernment>0784</localGovernment>
          <registered>1999-09-19T19:09:10</registered>
          <purpose>001</purpose>
        </item>
        <item>
          <cadastralUnit>012:345:68</cadastralUnit>
          <cuLocation>Cross Rode Square 1B</cuLocation>
          <province>0037</province>
          <localGovernment>0784</localGovernment>
          <registered>1999-09-19T19:09:10</registered>
          <purpose>001</purpose>
        </item>
      </response>
    </ns4:cuAdresResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## 7.1.9 Service with a MIME-attachment

### Query:

```

POST /cgi-bin/consumer_proxy HTTP/1.0
User-Agent: PEAR-SOAP 0.7.5-devel
Host: 195.50.218.64
Content-Type: multipart/related; type=text/xml; boundary="=_b5a8d09eeeb161be29def84633d6f6fc"
Content-Length: 1894

```

SOAPAction: ""

--=\_b5a8d09eeeb161be29def84633d6f6fc

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

Content-Transfer-Encoding:8bit

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

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

<SOAP-ENV:Header xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">

<xrd:consumer>assert</xrd:consumer>

<xrd:producer>land-cadastre</xrd:producer>

<xrd:userId>EE:PIN:ABC4567</xrd:userId>

<xrd:id>6cae248568b3db7e97ff784673a4d38c5906bee0</xrd:id>

<xrd:service>land-cadastre.uploadMime.v1</xrd:service>

<xrd:issue>Y-305-1</xrd:issue>

<xrd:position>engineer</xrd:position>

</SOAP-ENV:Header>

<SOAP-ENV:Body>

<ns4:uploadMime xmlns:ns4="http://assert.ee.x-rd.net/producer/">

<request href="cid:b8fdc418df27ba3095a2d21b7be6d802"/>

</ns4:uploadMime>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

--=\_b5a8d09eeeb161be29def84633d6f6fc

Content-Disposition:php5hmCiX

Content-Type:{http://www.w3.org/2001/XMLSchema}hexBinary

Content-Transfer-Encoding:base64

Content-ID:<b8fdc418df27ba3095a2d21b7be6d802>

a29ybmKJa3cNCmtvcnNpa2EJY28NCmtyZWVrYQllbA0Ka3JpaQn1ZA0Ka3Vt9Wtp  
CfZzDQprdXJkaQlrdQ0Ka3ZhbmpbhWEJa2oNCmv1bXJpCWN5DQpsYWRpbmEJbGEN  
Cmx1YmEJ9XINCmx1Z2FuZGEJ9XANcmz1dW5hbmRIYmVsZQlucg0KbOR0aQlSdg0K  
bWFrZWRvb25pYQltaw0KbWFsYWdhc3NpCW1nDQptb2xkb3ZhcW1vDQptb25nb2xp  
CW1uDQptdXN0bGFza2VlbAn2dQ0KbeRua3NpCWd2DQpuYW5haQnkcw0KbmF1cnUJ  
bmENCm5hdmFobwIudg0KbmRIYmVsZQn1cw0KbmRvbmdhCW5nDQpuZWVuzXRzaQn2  
dg0KbmVwYWxpCW5lDQpuaXZoaQnkdA0KbmbpT+YQlueQ0Kbm9nYWkj9ncNCg==

--=\_b5a8d09eeeb161be29def84633d6f6fc

/cgi-bin/consumer\_proxy HTTP/1.1

Host: 195.101.102.103

Pragma: no-cache

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, \*/\*

## Response:

HTTP/1.1 200 OK

Date: Tue, 02 Dec 2003 11:29:18 GMT

Server: Apache/1.3.27 (Unix) PHP/4.3.2RC4 mod\_ssl/2.8.14 OpenSSL/0.9.6j

X-Powered-By: PHP/4.3.2RC4

Status: 200 OK

Content-Length: 1653

Content-Type: multipart/related; type=text/xml; boundary="=\_9d665408f43f4698f71029c2df2b834e"

X-Cache: MISS from 195.101.102.103

Connection: close

--=\_9d665408f43f4698f71029c2df2b834e

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

Content-Transfer-Encoding:8bit

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  <SOAP-ENV:Header xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
    <xrd:consumer>assert</xrd:consumer>
    <xrd:producer>land-cadastre</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>6cae248568b3db7e97ff784673a4d38c5906bee0</xrd:id>
    <xrd:service>land-cadastre.uploadMime.v1</xrd:service>
    <xrd:issue>Y-305-1</xrd:issue>
    <xrd:position>engineer</xrd:position>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns4:uploadMimeResponse
      xmlns:ns4="http://assert.ee.x-rd.net/producer/">
      <request>52ed0ffbf27fc34759dce05d0e7bed2302876cec</request>
      <response>
        <answer>-1</answer>
        <otherInfo href="cid:793340a8216da081f3d785bcc74c0f74"/>
      </response>
    </ns4:uploadMimeResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
---_9d665408f43f4698f71029c2df2b834e
Content-Disposition:uploadfile.txt
Content-Type:application/octet-stream
Content-Transfer-Encoding:base64
Content-ID:<793340a8216da081f3d785bcc74c0f74>
```

```
VMO1ZWxpbmUgdGFsdiB2w7VpYiBzYWFiZWRRhIGhpGplbSBrdWkgdGF2YWxpc2Vs
dC4K
---_9d665408f43f4698f71029c2df2b834e
```

## 7.1.10 Standard error with header

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns4="http://assert.ee.x-rd.net/producer/"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd">
  <SOAP-ENV:Header>
    <xrd:consumer>abc</xrd:consumer>
    <xrd:producer>assert</xrd:producer>
    <xrd:userId>EE:PIN:ABC4567</xrd:userId>
    <xrd:id>3e47be25da43528deb639f5965d58b4d21bbc173</xrd:id>
    <xrd:service>assert.paring1.v2</xrd:service>
    <xrd:issue/>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>Server.InternalError</faultcode>
      <faultstring>Server internal error</faultstring>
      <faultactor>actor://myFunction</faultactor>
      <detail>
        <ns4:faultdetail>malloc failure</ns4:faultdetail>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```

    </detail>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## 7.2 SAMPLE SERVICE DESCRIPTIONS (WSDL)

### 7.2.1 Sample description of database services

```

<?xml version="1.0"?>
<definitions name="mydef"
  targetNamespace="http://land-cadastre.ee.x-rd.net/producer"
  xmlns:tns="http://land-cadastre.ee.x-rd.net/producer"
  xmlns:xrd="http://x-rd.net/xsd/xroad.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/">
<types>
  <schema
    targetNamespace="http://land-cadastre.ee.x-rd.net/producer"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://x-rd.net/xsd/xroad.xsd" />
    <element name = "cu">
      <complexType>
        <sequence>
          <element name="request">
            <complexType>
              <all>
                <element name="cadastralUnit" type="string">
                  <annotation>
                    <appinfo>
                      <xrd:wildcard>*?</xrd:wildcard>
                      <xrd:title>Cadastral unit indication</xrd:title>
                    </appinfo>
                  </annotation>
                </element>
                <element name="cuMax" minOccurs="0" type="tns:t_cuMax">
                  <annotation>
                    <appinfo>
                      <xrd:title>Maximum count of answers. By default 10</xrd:title>
                    </appinfo>
                  </annotation>
                </element>
              </all>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
    <element name = "cuResponse">
      <complexType>
        <sequence>
          <element name="request">

```

```

<complexType>
  <all>
    <element name="cadastralUnit" type="string"/>
    <element name="cuMax" minOccurs="0" type="tns:t_cuMax"/>
  </all>
</complexType>
</element>
<element name="response">
  <complexType>
    <sequence>
      <element name="item" minOccurs="0" maxOccurs="unbounded">
        <annotation>
          <appinfo>
            <xrd:title>Data of cadastral unit</xrd:title>
          </appinfo>
        </annotation>
        <complexType>
          <all>
            <element name="cadastralUnit" type="string">
              <annotation>
                <appinfo>
                  <xrd:title>Cadastral unit</xrd:title>
                </appinfo>
              </annotation>
            </element>
            <element name="cuOfficialName" minOccurs="0" type="string">
              <annotation>
                <appinfo>
                  <xrd:title>Official name of cadastral unit</xrd:title>
                </appinfo>
              </annotation>
            </element>
            <element name="registered" type="dateTime">
              <annotation>
                <appinfo>
                  <xrd:title>Registration date</xrd:title>
                </appinfo>
              </annotation>
            </element>
            <element name="purpose" minOccurs="0" type="tns:t_purpose">
              <annotation>
                <appinfo>
                  <xrd:title>Technical code of first purpose</xrd:title>
                </appinfo>
              </annotation>
            </element>
          </all>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
<simpleType name="t_purpose">

```

```

<annotation>
  <appinfo>
    <xrd:title>Type of purpose</xrd:title>
  </appinfo>
</annotation>
<restriction base="string">
  <enumeration value="001">
    <annotation>
      <appinfo>
        <xrd:title>(1)Residential land (E)</xrd:title>
      </appinfo>
    </annotation>
  </enumeration>
  <enumeration value="0010">
    <annotation>
      <appinfo>
        <xrd:title>(010)Land under nature preservation (H)</xrd:title>
      </appinfo>
    </annotation>
  </enumeration>
</restriction>
</simpleType>
<simpleType name="t_cuMax">
  <annotation>
    <appinfo>
      <xrd:title>Maximum count of answers</xrd:title>
    </appinfo>
  </annotation>
  <restriction base="string">
    <enumeration value="10">
      <annotation>
        <appinfo>
          <xrd:title>10</xrd:title>
        </appinfo>
      </annotation>
    </enumeration>
    <enumeration value="100">
      <annotation>
        <appinfo>
          <xrd:title>100</xrd:title>
        </appinfo>
      </annotation>
    </enumeration>
  </restriction>
</simpleType>
<element name = "legacy1">
  <complexType>
    <sequence>
      <element name="request">
        <complexType>
          <sequence>
            <element name="item"
              type="string" minOccurs="0" maxOccurs="unbounded">
              <annotation>
                <appinfo>
                  <xrd:title>Allowed method name for user</xrd:title>
                </appinfo>
              </annotation>
            </element>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>

```

```

        </appinfo>
      </annotation>
    </element>
  </sequence>
</complexType>
</element>
</sequence>
</complexType>
</element>
<element name = "legacy1Response">
  <complexType>
    <sequence>
      <element name="request">
        <complexType>
          <sequence>
            <element name="item"
              type="string" minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
      <element name="response">
        <complexType>
          <sequence>
            <element name="url" type="xrd:url">
              <annotation>
                <appinfo>
                  <xrd:title>Entry URL of information system</xrd:title>
                </appinfo>
              </annotation>
            </element>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
<element name = "uploadMime">
  <complexType>
    <sequence>
      <element name="request">
        <simpleType>
          <restriction base="hexBinary" />
          <annotation>
            <appinfo>
              <xrd:title>Data</xrd:title>
            </appinfo>
          </annotation>
        </simpleType>
      </element>
    </sequence>
  </complexType>
</element>
<element name = "uploadMimeResponse">
  <complexType>
    <sequence>
      <element name="request">

```

```

    <simpleType>
      <restriction base="hexBinary" />
      <annotation>
        <appinfo>
          <xrd:title>Data</xrd:title>
        </appinfo>
      </annotation>
    </simpleType>
  </element>
<element name="response">
  <complexType>
    <annotation>
      <appinfo>
        <xrd:title>Answer</xrd:title>
      </appinfo>
    </annotation>
    <all>
      <element name="answer" type="string">
        <annotation>
          <appinfo>
            <xrd:title>Explanation</xrd:title>
          </appinfo>
        </annotation>
      </element>
      <element name="otherInfo" type="base64Binary">
        <annotation>
          <appinfo>
            <xrd:title>Desired information</xrd:title>
          </appinfo>
        </annotation>
      </element>
    </all>
  </complexType>
</element>
</sequence>
</complexType>
</element>
</schema>
</types>
<portType name="myporttype">
  <operation name="cu">
    <documentation>
      <xrd:title>Cadastral unit request by register number</xrd:title>
      <xrd:title xml:lang="et">Katastriüksuse päring tunnuse järgi</xrd:title>
      <xrd:title xml:lang="en">Cadastral unit request by register number</xrd:title>
      <xrd:notes>Returns data entered to registry</xrd:notes>
    </documentation>
    <input message="tns:cu" />
    <output message="tns:cuResponse" />
  </operation>
  <operation name="legacy1">
    <documentation>
      <xrd:title>Enter to other information system</xrd:title>
    </documentation>
    <input message="tns:legacy1" />
    <output message="tns:legacy1Response" />
  </operation>
</portType>

```

```

</operation>
<operation name="uploadMime">
  <documentation>
    <xrd:title>MIME attachment method</xrd:title>
  </documentation>
  <input message="tns:uploadMime" />
  <output message="tns:uploadMimeResponse" />
</operation>
</portType>
<binding name="mybinding" type="tns:myporttype">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="cu">
    <xrd:version>v1</xrd:version>
    <soap:operation soapAction="" />
    <input>
      <soap:body use="literal"
        namespace="http://assert.ee.x-rd.net/producer/" />
      <soap:header message="tns:improvedHeader" part="*" use="literal"
        namespace="http://x-rd.net/xsd/xroad.xsd" />
    </input>
    <output>
      <soap:body use="literal"
        namespace="http://assert.ee.x-rd.net/producer/" />
      <soap:header message="tns:improvedHeader" part="*" use="literal"
        namespace="http://x-rd.net/xsd/xroad.xsd" />
    </output>
  </operation>
  <operation name="legacy1">
    <xrd:version>v1</xrd:version>
    <soap:operation soapAction="" />
    <input>
      <soap:body use="literal"
        namespace="http://assert.ee.x-rd.net/producer/" />
      <soap:header message="tns:standardHeader" part="*" use="literal"
        namespace="http://x-rd.net/xsd/xroad.xsd" />
    </input>
    <output>
      <soap:body use="literal"
        namespace="http://assert.ee.x-rd.net/producer/" />
      <soap:header message="tns:standardHeader" part="*" use="literal"
        namespace="http://x-rd.net/xsd/xroad.xsd" />
    </output>
  </operation>
  <operation name="uploadMime">
    <xrd:version>v1</xrd:version>
    <soap:operation soapAction="" style="document" />
    <input>
      <mime:multipartRelated>
        <mime:part>
          <soap:body parts="request" use="literal"
            namespace="http://assert.ee.x-rd.net/producer/" />
          <soap:header message="tns:standardHeader" part="*" use="literal"
            namespace="http://x-rd.net/xsd/xroad.xsd" />
        </mime:part>
        <mime:part>
          <mime:content part="p1" type="application/binary" />
        </mime:part>
      </mime:multipartRelated>
    </input>
  </operation>

```

```

    </mime:part>
  </mime:multipartRelated>
</input>
<output>
  <mime:multipartRelated>
    <mime:part>
      <soap:body parts="response" use="literal"
        namespace="http://assert.ee.x-rd.net/producer/" />
      <soap:header message="tns:standardHeader" part="*" use="literal"
        namespace="http://x-rd.net/xsd/xroad.xsd" />
    </mime:part>
    <mime:part>
      <mime:content part="p2" type="application/binary" />
    </mime:part>
  </mime:multipartRelated>
</output>
</operation>
</binding>
<message name="cu">
  <part name="request" element="tns:cu" />
</message>
<message name="cuResponse">
  <part name="response" element="tns:cuResponse" />
</message>
<message name="legacy1">
  <part name="request" element="tns:legacy1" />
</message>
<message name="legacy1Response">
  <part name="response" element="tns:legacy1Response" />
</message>
<message name="uploadMime">
  <part name="request" element="tns:uploadMime" />
  <part name="p1" type="base64Binary" />
</message>
<message name="uploadMimeResponse">
  <part name="response" element="tns:uploadMimeResponse" />
  <part name="p2" element="base64Binary" />
</message>
<message name="standardHeader">
  <part name="consumer" element="xrd:consumer" />
  <part name="producer" element="xrd:producer" />
  <part name="userId" element="xrd:userId" />
  <part name="id" element="xrd:id" />
  <part name="service" element="xrd:service" />
  <part name="issue" element="xrd:issue" />
</message>
<message name="improvedHeader">
  <part name="consumer" element="xrd:consumer" />
  <part name="producer" element="xrd:producer" />
  <part name="userId" element="xrd:userId" />
  <part name="id" element="xrd:id" />
  <part name="service" element="xrd:service" />
  <part name="issue" element="xrd:issue" />
  <part name="authenticator" element="xrd:authenticator" />
</message>
<service name="myservice">

```

```
<port name="myport" binding="tns:mybinding">
  <soap:address location="http://secureproxy/cgi-bin/consumer_proxy"/>
  <xrd:title>Land Cadastre</xrd:title>
  <xrd:title xml:lang="en">Land Cadastre</xrd:title>
  <xrd:address producer="land-cadastre" />
</port>
</service>
</definitions>
```