SHS Version 2.0 SOAP-based Protocol Binding to SHS Concepts

Försäkringskassan - Swedish Social Insurance Agency

Contents

1 Introduction

The purpose of this document is to describe the SOAP-based protocols that are used between SHS nodes and connected business applications in such details that this specification may be used as:

- input to implementation of SHS compliant products
- test specifications

1.1 Audience

This document is primarily intended for IT and product architects and specialists that need to implement, verify or put the SHS implementation in an overall IT architecture definition. The reader is assumed to have basic knowledge of HTTP, the SHS concept and Web Service standards.

1.2 Documentation structure

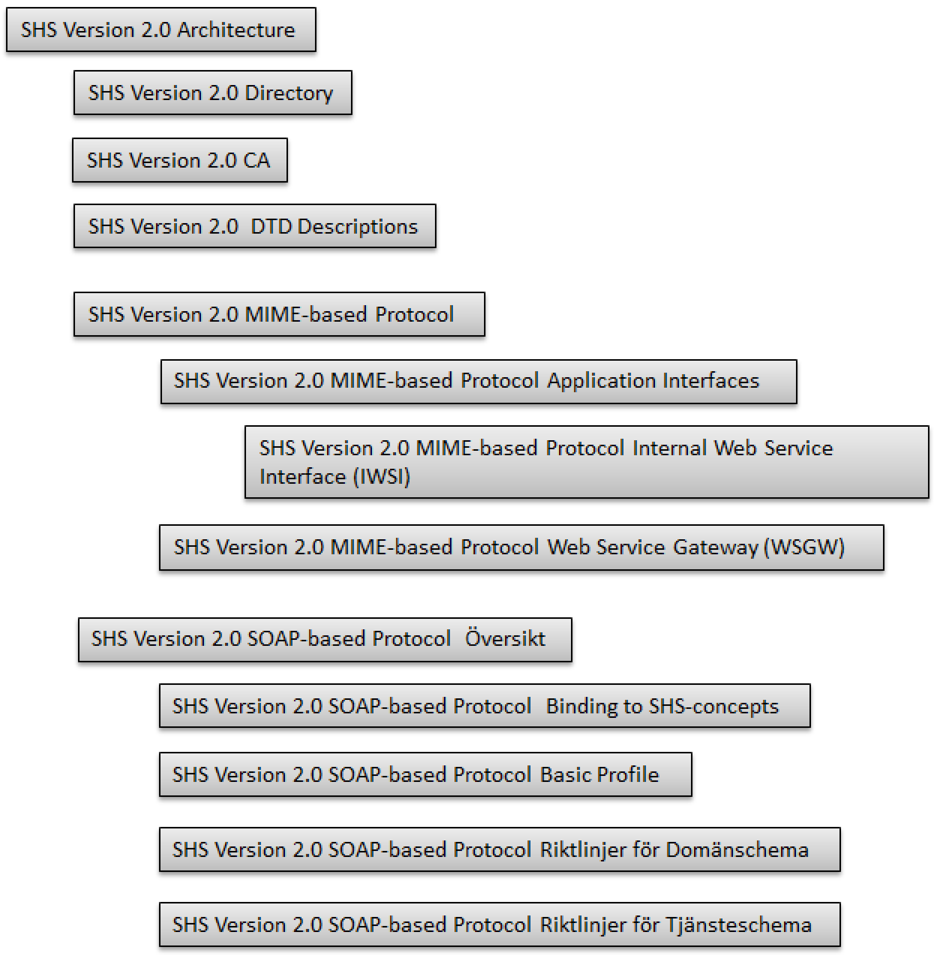
SHS Version 2.0 Architecture
SHS Version 2.0 Directory
SHS Version 2.0 CA
SHS Version 2.0 DTD Descriptions
SHS Version 2.0 MIME-based Protocol
SHS Version 2.0 MIME-based Protocol Application Interfaces
SHS Version 2.0 MIME-based Protocol Internal Web Service Interface (IWSI)
SHS Version 2.0 MIME-based Protocol Web Service Gateway (WSGW)
SHS Version 2.0 SOAP-based Protocol Översikt
SHS Version 2.0 SOAP-based Protocol Binding to SHS-concepts
SHS Version 2.0 SOAP-based Protocol Basic Profile
SHS Version 2.0 SOAP-based Protocol Riktlinjer för Domänschema
SHS Version 2.0 SOAP-based Protocol Riktlinjer för Tjänsteschema

Figure 1 Documentation structure overview

1.3 Documentation References

[API]	SHS Version 2.0 MIME-based Protocol Application Interfaces
[ARC]	SHS Version 2.0 Architecture

[BIND]	SHS Version 2.0 SOAP-based Protocol Binding to SHS Concepts
[BP]	SHS Version 2.0 SOAP-based Protocol Basic Profile
[CA]	SHS Version 2.0 CA
[DIR]	SHS Version 2.0 Directory
[DTD]	SHS Version 2.0 DTD Descriptions
[GD]	SHS Version 2.0 SOAP-based Protocol Riktlinjer för domänschema
[GT]	SHS Version 2.0 SOAP-based Protocol Riktlinjer för tjänsteschema
[IWSI]	SHS Version 2.0 MIME-based Protocol Internal Web Service Interface (IWSI)
[MIME]	SHS Version 2.0 MIME-based Protocol
[SOAP]	SHS Version 2.0 SOAP-based Protocol Översikt
[WSGW]	SHS Version 2.0 MIME-based Protocol Web Service Gateway (WSGW)

1.4 Document history

Version	Date	Change	By	Approved
0.1	2012-08-21	Document created		
	2012-10-12	SHS version 2.0, after 1'st submission for comment		
	2013-01-29	First complete version		
	2013-02-08	Final version SHS 2.0		

1.5 Requirements

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

1.6 Document Conventions

In this document the following conventions are used:

Times New Roman	Normal text
[REFERENCE]	References are normal text (UPPER CASE) enclosed in square brackets.
`Courier 10pt`	Technical details such as message examples
NoteSide	Margin notes that indicates examples or definitions

2 Overview

In order to provide interoperability between the SHS systems from different suppliers, it is necessary to standardise the network and transport protocols and corresponding transport formats to be used by SHS.
This document describes the SOAP-based SHS Protocol and mainly it's binding to the SHS concepts. Detailed information and guidelines regarding the message format are in [SOAP], [BP], [GT] and [GD].

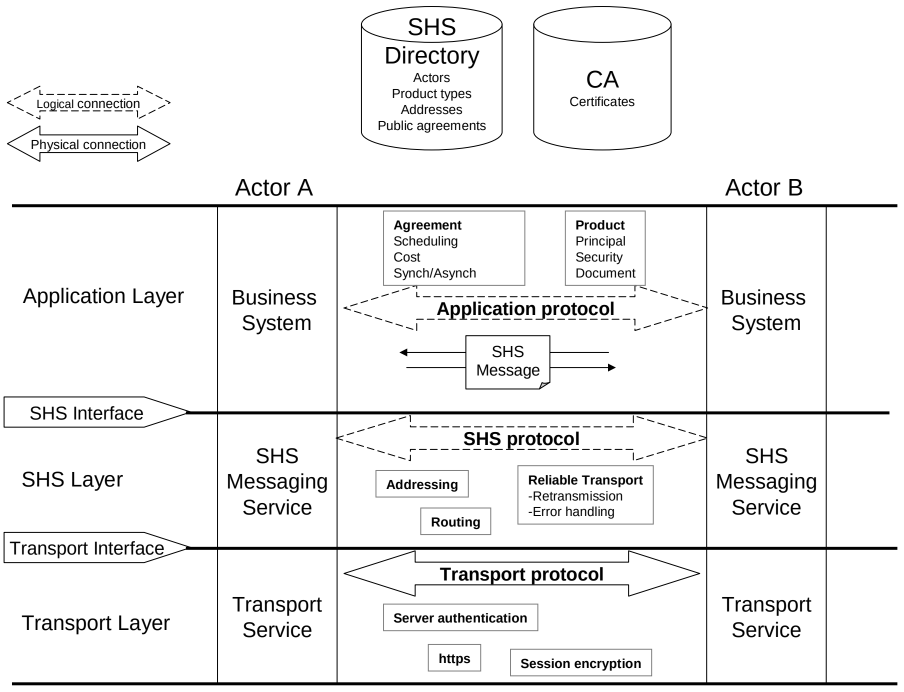

Figure 2 SHS communication layers

There are different layers and different entities involved in a message transmission between two business systems. The following are the major interaction layers defined in SHS:

1. Transport layer based on HTTP/SSL
2. SHS layer
3. Application layer

2.1 Transport layer

The transport layer defines how SHS uses the underlying HTTP/SSL interaction, and the binding specification is SOAP 1.1 HTTP binding and WS-I Basic profile, see [BP].
The security function includes strong authentication of communicating parties (client identification) and transport encryption using SSL/TLS.

2.2 SHS layer

The SHS layer may be described as the aspects of the communication that are actually fully controlled within the SHS implementation and includes functions as: agreement handling, directory access, error handling and address routing.
The core of SHS that handle the actual message transport and routing decisions are referred to as SHS Messaging Service.
The SHS layer aspects are described in more detail in Chapter 5 and 6.

2.3 Application layer

Application layer refers to the information exchange that SHS enable between two business applications.
Business applications that use SHS may exchange information according to one of three service interaction scenarios.

- Request-response (Fråga-svar)
- Oneway (Informationsspridning)
- Request-reply (Uppdrag-resultat)

The interaction between two end business systems is defined by the agreement and its corresponding product type description. The involved messages are seen as standard SHS messages, which mean that SHS is only acting as a deliverer of these messages.
The application layer protocols are not found in methods or specific commands between nodes but rather in message specifications that govern how SHS will handle the message. Examples of such information is

- addresses
- product type information

3 Protocols

The SHS SOAP-based protocol is an application of SOAP 1.1 according to WS-I Basic Profile. As only synchronous communication is supported only the receive service (RS) is affected.
This and the following chapters describe how the receive service for the SOAP-based protocol is implemented in detail.
The parameters that govern the SHS SOAP-based protocols are

- The service interaction type for the communication, i.e. oneway (informationsspridning), request-response (fråga-svar) and request-reply (uppdrag-resultat)
- The product type is derived by a mapping from some information in the SOAP envelope of the delivered the message or found in the message header
- Sender address
- Receiver address
- etc

Whereas the initiating business application can use the synchronous receive service there is currently no defined interface between SHS and the responding application (the server end) in the synchronous mode. Instead this is up to implementers to

realize in the form of a plug-in or equivalent modular component. The format of a synchronous response SHALL be based on the SOAP message syntax.
Where the MIME-based protocols offer plenty addressing scenarios only two are supported by the SOAP-based protocols. These are direct addressing and implicit addressing.

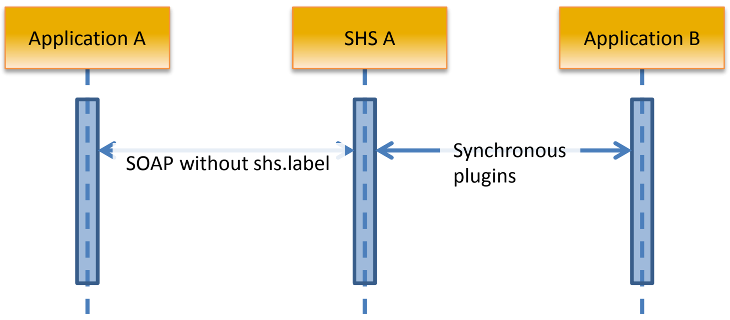

Figure 3 Implicit addressing scenario

In the implicit addressing scenario the client does not supplies any addressing information (no shs.label element or no soap:header at all). The SHS node interprets this as a call directed to itself.

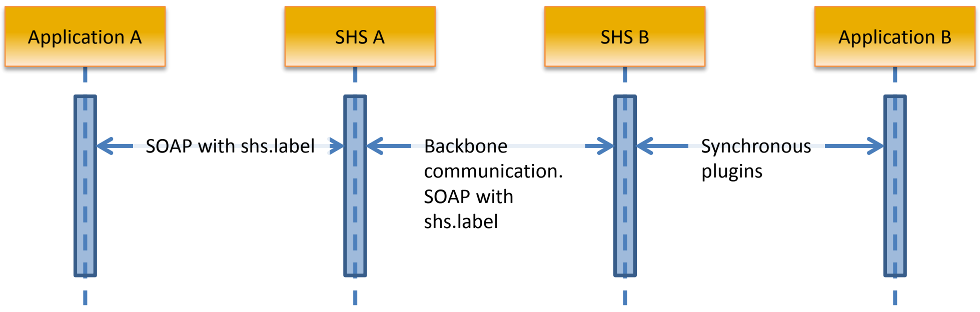

Figure 4 Direct addressing scenario (with routing)

The client supply a shs.label element within the soap:header. If the message is addressed to an external node it is routed to the other node by means of the backbone communication protocol.

4 Transport layer protocols

4.1 General communication considerations

Use of HTTP MUST be in conformance with SOAP 1.1 and WS-I Basic profile 1.1. Generally HTTPS is used. For internal message access HTTP MAY be used depending on internal security policy.
The connections that are established use either HTTP or HTTPS. The minimum key length that SHOULD be used with HTTPS is at least 1024 bit for asymmetric keys and 128 bits for symmetric keys.
Implementations SHALL support SSL and SHOULD support TLS.

4.2 Port number conventions

SHS installations are recommended to use ports 443, 8083 or 11288 for SHS backbone communication

5 SHS layer protocols

The SHS layer may be viewed as the middleware that enables the application layer needs for a transparent transfer of information between businesses applications with the mechanism made available by the HTTP methods (the transport layer). In order to provide this the SHS layer is responsible for:

- Synchronous transfer based service interaction scenarios
- Routing of SHS messages
- Exchange of agreements
- Error handling and creation of log entries.
- Reliable transfer of messages
- Tracing of messages

SHS messaging service implements the “services” of the SHS layer in an SHS node. The messaging service interprets transport error codes and handles error messages.

5.1 The message handling process (receive service)

The diagram in figure 5 outlines the SHS messaging service operation for the SOAP-based protocols with respect to:

1. An incoming message
2. Message parsing and collection of message metadata
3. Checks and processing done at the receiving SHS-node.
4. Making an outbound call to another node (routing) or calling the internal plugin system
5. Collection end return of result
6. Error handling

Transport-level parts of message handling including use of SSL/TLS are governed by the SOAP binding to HTTP. The following section describes message handling at the SHS level.

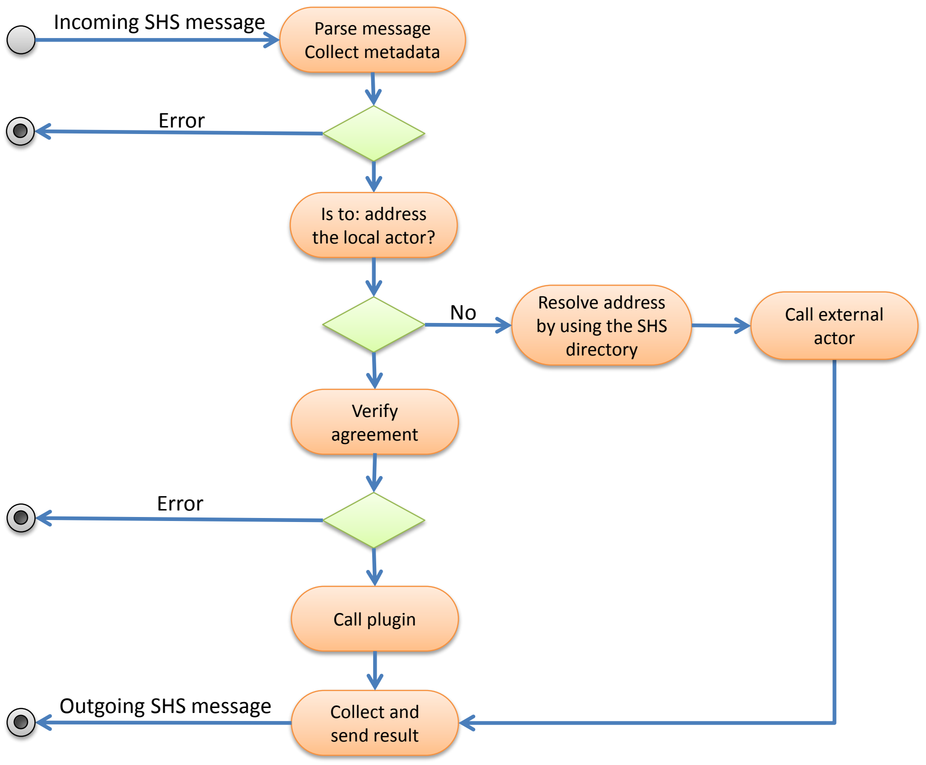

Figure 5 Message handling process

5.1.1 Parse message and collect metadata

The incoming message is parsed and its structure verrified as a valid SOAP-message and if a soap:Header with a shs.label element exists it has follow SHS rules. Metadata in the header are collected for logging and further use in the process. A couple of things can happen in this stage:

1. The message is found invalid. A malformed syntax error is thrown
2. The message do not have a soap:Header or a header without a shs.label element (Implicit addressing) meaning no metadata can be extracted from the header. Necessary metadata has to be obtained another way.
    a. Receiver address is implied to be the local actor
    b. Sender address is extracted from the certificate used for identification when the SSL/TLS session is established. The means by which this is done is outside the scope of this specification, e.g. extracted by a servlet-filter and propagated forward with the message.
    c. Producttype is derived by a mapping (key-lookup) from the unique service name, that is the root element of the soap:Body if necessary including it’s namespace.
3. A soap:Header with a shs-label element is found. Receiver address and Sender address are extracted from the header.
    a. Producttype is derived by a mapping (key-lookup) from the unique service name. That is the root element of the soap:Body if necessary including it’s namespace…

b. …or if a product element is present in the header it is used as is (this shold really only happen when receiving a message in backbone communication)
c. If a from-address is found the value is checked against a part of the certificate used for authentication. E.g. the whole or a part of DN or CN.
d. Receiver address can be either an internal one (local) or an external one. The node determines by a local configuration lookup if the address is internal or external.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tan="http://fk.se/SHS/xsd/tanden">
     <soapenv:Body>
      <tan:FK.TV.TestRoundTripRequest
      organization-number="1234567898"
      request-id="a4924e42-4214-ba93-0014-98739237ba27"
      shs-invoked-interface="Submit Claim v2"
      vendor-name="Klinik X"
      product-name="Undersökning Y"
      version-number="1"
      user-id="197701011234">
         <tan:clinic-id>33312341</tan:clinic-id>
      </tan:FK.TV.TestRoundTripRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 6 Product type mapping

As shown in figure 6, product type is mapped either from :
1. Root element of body
2. Root element of body including namespace

5.1.2 Internal call handling

An internal call is handled in the local node. The local node must have a suitable agreement to be able to handle the message. Othewise a missing agreement error is thrown.
Exactly how the node carries out the procedure to obtain an answer is outside the scope of this description. It is left to the implementer to decide. Some sort of configurable or rule-driven plugin mechanisms are assumed.

5.1.3 External call handling (routing)

An SHS node routes a message according to information found in the SHS message label.

Addresses are constructed according to a pattern described in reference [DTD]. Following this pattern ensures that an address is a globally unique identifier of an SHS node, an internal business system, a service etc.
Adressing mode for the SOAP-based protocol is always direct addressing. To resolve the logical Receiver address the node does a lookup in the global SHS directory with the (logical) address as key and a network address (an URL) is returned.
The message header (shs.label) is copied to the outgoing message and updated with a timestamp (datetime) and a transaction ID (tx.id). The SHS transaction ID MUST be unique and only used once. In case of an answer with shs.label the same transaction ID as in the request should be used.

5.1.4 Error handling in the message flow

All faults in the technical message flow (mainly node-to-node communication) MUST be returned as SOAP-faults.
Application errors could be returned as SOAP Faults or be handled at the application layer by other means.
In the case of an error the node MUST return http-status 500 and SHOULD return a SOAP Fault in conformance with SOAP 1.1 http binding.
A SHS specific fault structure MUST be placed inside the detail element. An example:

```
<shs:fault-data>
      <shs:tx-id>72621291-9e62-f598-423c-7fee481ac1d2</shs:tx-id>
      <shs:error-code>MissingAgreement</shs:error-code>
      <shs:description>
            Agreement missing at local delivery
      </shs:description>
</shs:fault-data>
```

SHS Error code	Meaning (SV)	Meaning (EN)
UnresolvedReceiver	Mottagare kan ej fastställas	Unable to resolve reciever
MissingAgreement	Överenskommelse saknas vid lokalt uttag	Agreement missing at local delivery
MissingDeliveryAddress	Misslyckad uppslagning av inlämningsadress	Delivery address not found in directory
MissingDeliveryExecution	Misslyckad leverans av meddelande	Delivery of message failed
IllegalProductType	Otillåtet värde på produkttyp	Illegal product type
UnknownProductType	Okänd produkttyp	Unknown product type
IllegalReceiver	Otillåtet värde på mottagande aktör	Illegal syntax for receiving actor
UnknownReceiver	Okänd mottagande aktör	Receiving actor unknown
IllegalSender	Otillåtet värde på avsändande aktör	Illegal syntax for sending actor
UnknownSender	Okänd avsändande aktör	Sending actor unknown
IllegalMessageStructure	Otillåten struktur på SHS-meddelandet	Illegal Message Structure
OtherError	Annat oväntat fel	Ohter error

5.1.5 SOAP Service interactions and SHS transfertypes

The three possible service interactions are described in [OV]. All service interactions are bound to the synchronous transfer type in SHS. Synchronous communication is characterised by blocking of the calling application until either a result or an error message is returned. The Messaging Service manage a chain of calls between applications.
All defined service interactions are synchronous in a SHS perspective. From a business perspective asynchronous services can be built upon the primitive synchronous services as is shown by the request-reply interaction.

6 Message format

Message format MUST conform to SOAP 1.1, WS-I Basic profile, see [BP], and the guidelines given in [GT] and [GD].
Client direct addressing and the SHS backbone communication require some header information. That information is placed in an element, shs-label, contained in the soap:Header element.

6.1 The label

The label, see Appendix A, contains control information for the SHS-message. It includes:

- Sender address (from element)
- Sender address type (addresstype attribute)
  - Only one addresstype is initially defined - ORGNR
- Receiver address (to element)
  - Only one addresstype is initially defined - ORGNR
- Receiver address type (addresstype attribute)
- Product type (product element)
- Protocol version information (version attribute)
- Time stamp (datetime element)
- Unique identification of this message interaction (tx-id attribute)
- Correlation id for relating requests and responses (corr-id attribute)
- Attribute mustUnderstand, if present, MUST be set to “0”

Example shs-label element:

```
<shs:shs-label version=”2.0” tx-id=”72621291-9e62-f598-423c-7fee481ac1d2”
corr-id="order-8897">
  <shs:to address-type="ORGNR">urn:X-shs:5563162469</shs:to>
  <shs:from address-type="ORGNR"> urn:X:shs:5563162469</shs:from>
  <shs:product>urn:X-shs: 72621291-9e62-f598-423c-
7fee481ac1d2</shs:product>
  <shs:datetime>2012-08-26T09:25:32</shs:datetime>
</shs:shs.label>
```

6.2 Shs-label Xml-schema

6.2.1 Shs-label

Shs.label is a part of the Xml-schema for shs. See appendix A.
The root element *shs-label* contains information of how the backbone communication should manage the content of a SHS message. A label is attached to every SHS message sent between nodes and it is placed in the soap:header.
If a message arriving to a node do not contain the shs.label element it is a case of implicit addressing and should be handled locally by the node.

6.2.1.1 Elements:

`from`	Specifies the actor that sent the message. The <shs-actor> production in [DTD] defines the syntax for the element value.
`to`	Specify the actor that is to receive the message. The <shs-actor> production in [DTD] defines the syntax for the element value.
`product`	Specifies the product type that the message contains. The element value contains a unique identity (URN) for the product type that the label is defined for. The <shs-product> production in [DTD] defines the syntax for the element value. The *product* element should in most cases be omitted in the call from a client to the first node.
`datetime`	The time stamp for the label creation according to ISO 8601 extended format 'yyyy-mm-ddThh:mm:ss'. This is the message time stamp. This attribute is set by the first node in a communication chain and SHOULD not be set by a client.

6.2.1.2 Attributes:

`version`	Specifies the version of SHS label that an instance of the shs.label root element conforms to. It is optional from client to node but MUST be used in communication between nodes.
`tx-id`	Specifies the uuid for a message transaction. Use is optional from client to node but required in exchange between nodes.
`corr-id`	specifies the correlation identity so that the sender application can associate any received response with a query. This attribute is optional.

6.2.2 From

From specifies the actor that sent the message.

6.2.2.1 *Attributes:*

`address-type`	Specifies the type of address used in the From element.

6.2.3 To

To specifies the actor that is the intended receiver of the message.

6.2.3.1 *Attributes:*

`address-type`	Specifies the type of address used in the To element.

7 SHS Directory access protocol

An SHS implementation MUST support LDAP for lookup of information in the global directory. This information includes actors, products, addresses and public agreements. The objects classes and attributes of directory entities are described in detail in [DIR].

There is not specific support in the directory for the MIME-based protocols or the SOAP-based protocol. It's not possible to decide which protocol to use by looking at information in the directory.

The attribute shsDeliveryMethods in an address object MUST point to an endpoint-url that can handle the intended protocol.

8 Application layer

The highest level, the application layer refers to the information exchange that SHS enable between two business applications. This includes information transfer as well as notification whether the data they received is good or bad and is not handled specifically by the SHS information structure.

8.1 Message flow

Message flow at the application layer are based on the service interaction types defined in [OV]. These message flow patterns are common ones in many use scenarios. More complex application layer scenarios can be constructed with these service interactions as building blocks.

The request-reply service interaction is composed of two independent oneway or request-response interactions. It is strongly recommended that a correlation id is used to maintain tracabilty through the nodes.

Appendix A: Xml-schema for SHS

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://schema.forsakringskassan.se/shs/2.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:shs="http://schema.forsakringskassan.se/shs/2.0"
           elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation xml:lang="en">
            Xml-Schema for shs-label and fault-data used in SHS 2.0 backbone
            and client communication protocol
        </xs:documentation>
    </xs:annotation>

    <xs:simpleType name="Address">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ORGNR"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="Actor">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="address-type" type="shs:Address" default="ORGNR"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:simpleType name="ErrorCode">
        <xs:restriction base="xs:string">
            <xs:enumeration value="UnresolvedReceiver"/>
            <xs:enumeration value="MissingAgreement"/>
            <xs:enumeration value="MissingDeliveryAddress"/>
            <xs:enumeration value="MissingDeliveryExecution"/>
            <xs:enumeration value="UnknownProductType"/>
            <xs:enumeration value="IllegalReceiver"/>
            <xs:enumeration value="UnknownReceiver"/>
            <xs:enumeration value="IllegalSender"/>
            <xs:enumeration value="UnknownSender"/>
            <xs:enumeration value="IllegalMessageStructure"/>
            <xs:enumeration value="OtherError"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="TxId">
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-
9a-fA-F]{12}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="Product">
        <xs:restriction base="xs:string">
            <xs:pattern value="urn:X\-shs:[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-
fA-F]{4}-[0-9a-fA-F]{12}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:element name="shs-label">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="from" type="shs:Actor"/>
                <xs:element name="to" type="shs:Actor"/>
                <xs:element name="datetime" type="xs:string" minOccurs="0"/>
                <xs:element name="product" type="shs:Product" minOccurs="0"/>
                <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
namespace="##other"/>
            </xs:sequence>
            <xs:attribute name="version" default="2.0" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="2.0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="tx-id" use="optional" type="shs:TxId"/>
            <xs:attribute name="corr-id" use="optional" type="xs:string"/>
        </xs:complexType>
    </xs:element>

    <xs:element name="fault-data">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="tx-id" minOccurs="0" type="shs:TxId"/>
                <xs:element name="error-code" type="shs:ErrorCode"/>
                <xs:element name="description" type="xs:string" minOccurs="0"/>
                <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"
namespace="##other"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```