SHS Version 1.2 Protocols

Verva - Swedish Administrative Development Agency

Editors:

Kurt Helenelund, Stephan Urdell, Bo Sehlberg,
Anders Bremsjö, Anders Lindgren, Jan Lundh, Christer Marklund

Contents

1 Introduction

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

- input to product and 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 and MIME standards.

1.2 References

[DTD]	SHS Version 1.2.01 DTD Descriptions
[API]	SHS Version 1.2.01 Application Interfaces
[ARCH]	SHS Version 1.2 Architecture
[DIR]	SHS Version 1.2.01 Directory
[CA]	SHS Version 1.2 CA
[HTTP]	HTTP 1.1 (Hypertext Transfer Protocol), RFC 2616
[MIME]	MIME 1.0 (Multipurpose Internet Mail Extensions) RFC 2045, 2046, 2047
[SMIME]	S/MIME (Security Multiparts for MIME) RFC 1847 and version 2 message specification RFC 2311

1.3 Document history

Version	Date	Change	By	Approved
0.1	2003-03-27	Initial structure based on information in SHS 1.1 API and DTD documentation	Anders Lindgren	
0.2	2003-04-30	First draft to serve as input to documentation workshop	Anders Bremsjö, Anders Lindgren	
0.3	2003-05-16	Updates after Luleå workshop	Anders Lindgren	
0.9	2003-09-22	Update based on comments from ÄR 2003-09-02	Christer Marklund, Anders Lindgren	
1.2	2003-10-09	Final version for SHS 1.2	Anders Lindgren	Jan Lundh

1.2.01-A	2004-05-14	First draft (A) of update for version 1.2.01 • Sort of message listings • Better support for duplicate control • Clarifying of error handling • Content ID in message listings	Anders Lindgren	
1.2.01-B	2004-06-03	Second draft for comments and approval	Anders Lindgren	
1.2.01-C	2004-11-22	Third draft including feedback • SHS Duplicate Message flag	Anders Lindgren	
1.2.01	2007-04-05	Published as Verva document, Final sub version	Anders Lingdren	Christer Marklund Jan Lundh

1.4 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.

1.5 Terms

The following terms are specific to this document.

RS	Receive service – service in SHS nodes that are called by connected applications or other SHS nodes to submit a message (or transfer to next SHS node)
DS	Delivery Service – service implemented in the SHS node which allows connected applications to fetch asynchronous messages
POST	HTTP POST method is used when the initiating part (client) sends information to a responder (server). Defined in [HTTP].
GET	HTTP GET method is used when the initiating part (client) retrieves information from a server. Defined in [HTTP].
MIME	SHS uses MIME message structures to encode structured, multipart messages. For example the SHS label and the payload data conveyed in separate body parts of a MIME message. Defined in [MIME]
S/MIME	S/MIME is an extension to MIME that allows signed and encrypted body parts. SHS uses this when end to end signing and encryption is needed.

Defined in [SMIME]

The following definitions are based on information from SHS Architecture document [ARCH] and are provided here for convenience only.

SHS Messaging Service - the component that executes the core services of SHS, which are routing, error handling logging etc.

SHS Node - an actual server implementation of the SHS messaging service.

SHS Network – the collection of SHS nodes that are interconnected in a network mesh.

SHS Actor – The SHS Actor is a part exchanging SHS messages with other actors. The actor has an SHS node (or accessing an SHS-node as a service) and has connected one or more of its business applications to the SHS network.

SHS Product Type – well defined structures of the artefacts transported in the SHS network, mainly various business documents. There is always an SHS actor who is the owner of a product type.

SHS Product – is an instantiated product type. One ore more products are packaged in an SHS message before submitted to SHS for transportation.

SHS Message – The structure SHS uses to exchange information. An SHS message can contain one or more SHS Products.

SHS Data Part – The smallest entity of information to be transferred by SHS. SHS data parts are packaged in an SHS message according to the product type definition.

SHS Agreement – a set of rules that governs the handling of a given product type between two actors. An agreement declares for example transfer cost, confirmation requirements and if a product is composed of a reply/request scenario. The principal (one of the actors) is the owner of the agreement.

SHS Address – the means by which a communicating party is referenced

SHS Directory – a globally available repository where the SHS messaging service may find information on other SHS Actors, SHS Products, SHS addresses and public agreements.

SHS Interface – methods that define how a messaging service can be utilized by an application or other service. The interface is implemented by a number of API's and client applications

Business system – any type of application that may need to exchange information with other systems or organisations, e.g. a tax administration system. The business system connects to SHS using one of the interfaces to SHS.

CA – the Certificate Authority is an authority that issues and manages security credentials and public keys. SHS uses external providers of these services.

1.6 How to read this document

This document describes different aspects of the SHS protocols.

- Chapter 2 is intended as an overview and also refers to the more detailed sections.
- Chapter 3 describes the protocol building blocks that are actually implemented as services in the SHS nodes.
- Chapter 4 describes how the building blocks combined provide the end-to-end services to connected business systems.
- Chapter 5-7 describes details of protocols and formats from a layered perspective.

1.7 Document Conventions

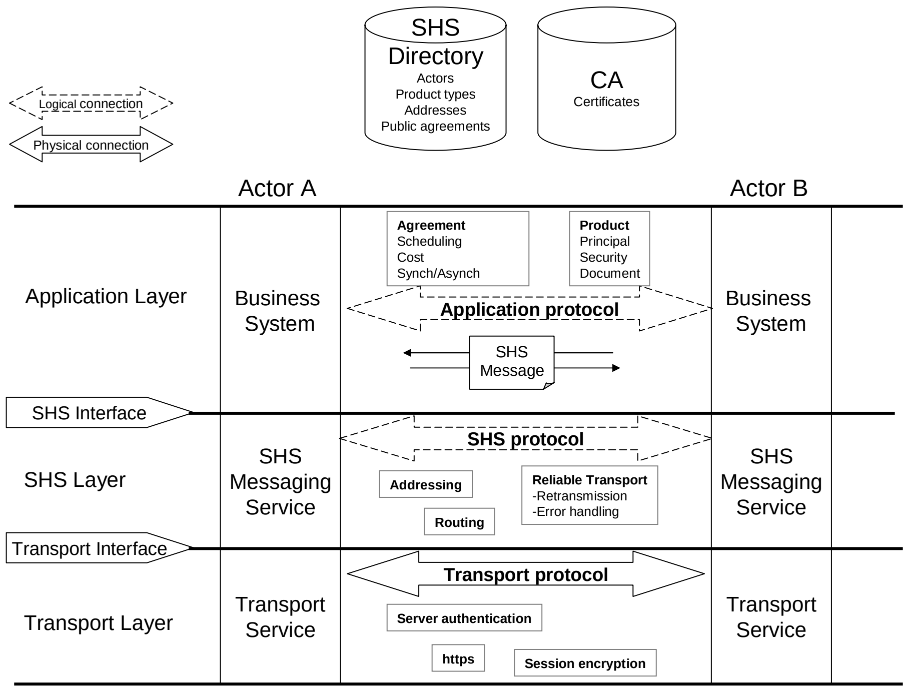
In this document the following conventions are used:

Times New Roman	Normal text
[REFERENCE]	References are normal text (UPPER CASE) enclosed in square brackets.
`Courier 10pt`	Technical details such as message examples
`Courier 8pt`	Same as 10pt courier when text length and content requires compact
NoteSide	Margin notes that indicates examples or definitions

Syntax descriptions are based on an augmented form of Backus Naur Format (BNF) as used in RFC2141. This form of BNF is used for not only URN descriptions but also to describe other syntax rules.

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.

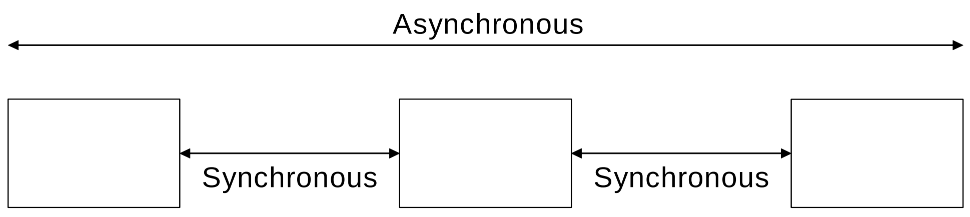


There are different layers and different entities involved in a message transmission between two business systems. The following are the major interaction layers defined in SHS:

1. Transport layer, HTTP/SSL
2. SHS layer
3. Application layer

Each layer is described in detail in the following sections.

Business applications that use SHS may exchange information in either synchronous or asynchronous patterns. The asynchronous capability is based on intelligence in the SHS messaging service. This functionality does not exist in the transport layer (HTTP) where all information exchange is synchronous.



SHS defines two major protocols:

- SHS Backbone Communication – the protocol between SHS nodes implemented using the receive service (RS). This service is used when a message is posted to an SHS node and for transfer between SHS nodes.

- SHS Internal Message Access – the protocol between a connected business application and an SHS server. This protocol is implemented using the delivery service (DS) that may be called when an application list or fetches incoming (asynchronous) messages and the receive service (RS) which is used when a message is posted to SHS.

2.1 Transport layer

The transport layer defines how SHS uses the underlying HTTP/SSL interaction, and could be seen as a binding specification.

SHS uses the POST and GET methods for its communication. SHS uses HTTP persistent connections (session reuse) and message formatting (MIME compliant headers).

In chapter 5 HTTP requirements is specified in more detail. Chapter 3 describes how HTTP MIME headers used in the actual "wire protocol".

The security function includes strong authentication of communicating parties and transport encryption using SSL.

2.2 SHS layer

The SHS layer may be describes as the aspects of the communication that are actually fully controlled within the SHS implementation and includes functions as: store and forward, error handling and routing.

The core of SHS that handle the actual message transport and routing decisions are referred to as SHS Messaging Service. The messages structures adhere to MIME standard [MIME]. When message data parts are either encrypted or signed the S/MIME [SMIME] structure is used.

The SHS layer aspects are described in more detail in Chapter 6.

2.3 Application layer

Application layer refers to the information exchange that SHS enable between two business applications.

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

- label attributes and elements such as transfer-type, sequence-type and meta-data
- product type information

A more complete description of properties that governs messages handling is found in section 7

2.4 Acknowledgements and Error Messages in SHS

When sending messages between systems, there is always a need to be able to determine whether a message has reached its destination or if there was a delivery problem. This is handled at several levels in SHS by protocol features and special messages being sent between systems.

There are three major levels, that can easily be distinguished, at which the need for this kind of information occurs (although other variants and finer sublevels can be found):

1. At low level where actual transmissions are performed between systems. The sending node needs to know whether the receiving node has received the message correctly. Aspects of this are covered in both transport layer (HTTP status) and in SHS layer (error messages).
2. When a message is sent through SHS, the sender will get an error message if the message can not be delivered as specified or the sender might wish to receive a confirmation that the message has reached its ultimate destination. The

error message is sent automatically by the SHS when the message can not be delivered as specified. The delivery confirmation is sent automatically if configured by the SHS when the receiving application has received the message by fetching it from its SHS, or by other means.

3. An application might send an application confirmation that it has processed the information received.

At all levels, an error condition will result in an error message being sent instead of a confirmation.

2.5 External Standards

SHS is based on a number of public specifications. The specifications are listed here with short descriptions, which version SHS uses and comments.

XML – eXtensible Markup Language. XML describes a class of data objects called XML documents and partially describes the behaviour of computer programs which process them.

Version used in SHS: 1.0

Comment: XML is primarily used to encode SHS specific information such as label and administrative messages

HTTP - Hypertext Transfer Protocol. It is a generic, stateless, protocol primarily used as a base for distributed object management in SHS.

Version used in SHS: 1.1

SSL - Secure Sockets Layer, is used by SHS for client and server authentication as well as encrypted connections.

Version used in SHS: 3.0

TLS - Transport Layer Security, defined in RFC 2246. TLS provides services comparable to SSL, namely client and server authentication and encrypted connections.

Version used in SHS: 1.0

Comment: Most implementations still use SSL “by tradition”. It’s up to SHS users to agree upon TLS or SSL.

LDAP - Lightweight Directory Access Protocol is used by SHS nodes for lookup in global directory of agreement, product and organisational information. It may also be used to find certificate information at the certificate authority (CA)

Version used in SHS: 3

Comment: No extensions of LDAP version 3 is required by SHS, which means that LDAP version 2 may optionally be used

MIME - Multipurpose Internet Mail Extensions is an important standard for SHS since the SHS message is based on the MIME syntax and structure

Version used in SHS: 1.0

S/MIME - Secure/Multipurpose Internet Mail Extensions is used in SHS when data parts are either signed or encrypted (or both) by SHS for end-to-end transmission security.

Version used in SHS: 2.0

X.509 – certificate format. SHS implementation SHALL support the X.509 version 3 certificate format. This is described in further detail in [CA].

Version used in SHS: 3.0

3 Protocol

SHS is based on two major protocols:

- SHS Backbone Communication
- SHS Internal message access

These protocols are implemented by the services receive service (RS) and delivery service (DS).

This chapter describes how these services are implemented in detail. Please see chapter 4 for how these are used in a system to provide the SHS end to end service (Messaging Service) and chapter 5-7 for a layered perspective.

The parameters that governs the SHS protocols are

- The sequence type[1] for the communication
- If the communication is synchronous or asynchronous (transfer-type)
- Which service that implements the responder. The services defined are

[1] Currently defined sequence types are: event, request, reply, adm

- Receive Service (RS) based on a request/response exchange pattern over the HTTP POST method
- Delivery Service (DS) based on a message response pattern over the HTTP GET method augmented by a status update based on the HTTP POST method

The following table show the combinations of protocols, transfer modes, the server implementations and HTTP binding.

Table 1 Protocols, transfer modes, implementation and bindings

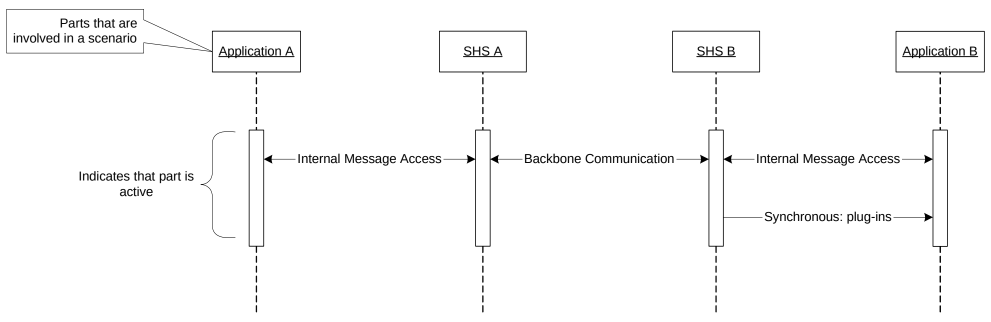
Protocol	**Mode**	**Server (responder) Implemented as**	**HTTP binding**
SHS Backbone Communication	Asynchronous	Receive Service (RS-async)	POST
	Synchronous	Receive Service (RS-sync)	POST
SHS Internal Message Access	Asynchronous	Receive Service (RS-async)	POST
		Delivery Service (DS-async)	GET (list/fetch message)
			POST (acknowledge message)
	Synchronous	Receive Service (RS-sync)	POST

The table shows that there are three different services that should be implemented in a SHS node, namely

- The asynchronous variant of the receive service (RS-async)
- The synchronous variant of the receive service (RS-sync)
- The asynchronous only delivery service (DS-async)

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 SHS message syntax.

In the following sections UML sequence diagram will be used to illustrate the protocols with respect to actors involved and timing requirements between actors.



3.1 Receive Service

When an SHS message is delivered to an SHS node from an application or other SHS node, this is realized through a receiving service in the receiving SHS node, using HTTP/SSL.

The receiving service has an URL <RSURL>, working as a base for all its communication.

The POST command SHOULD NOT use any options except the mandatory HTTP version, since they will be ignored by the server, however the keep-alive attribute MAY be used for backward compatibility according to RFC2616. This is further described under 5.1.

The entity header SHALL have a Content-Type of “message/rfc822”.

If the response contains status return codes these MUST be encoded as extension-header (defined in 3.1.1 and 3.1.2).

The transfer-type attribute in the label of the SHS message controls if the transfer is asynchronous or synchronous. The major difference between the transfer types is that asynchronous responses only contain a status or identifier (when the message first is submitted to SHS) whereas the synchronous response is either an error status or the requested information. The synchronous response SHALL be formatted using SHS message structure.

3.1.1 Asynchronous variant (RS-async)

An asynchronous transfer type means that no data/payload information is returned in the response. The response SHALL include extended headers as defined below to return status information about the transfer.

http header extenstion	Description
X-shs-txid	SHS transaction ID, a unique ID of the transaction created
X-shs-corrid	SHS correlation identity (text)
X-shs-contentid	SHS content identity (text)
X-shs-localid	A local trace ID assigned by the local SHS node
X-shs-nodeid	The SHS node ID
X-shs-arrivaldate	The time for the first succesful submission of a message with a given identity. If this time is in the past the messages have already been accepted by SHS and the current submmission is a duplicate.
X-shs-duplicatemsg	Set to “yes” if SHS node detects a duplicate message, otherwise “no”

The extended header MUST follow the format for extended entity headers generally defined in [HTTP] as:

```
shs-extension-header = extension-header
extension-header = message-header
message-header = field-name ":" [ field-value ]
```

The specific field/field-value combinations allowed for SHS is

Definition:

```
X-shs-txid = “X-shs-txid” “:” SHS transaction ID
X-shs-corrid = “X-shs-corrid” “:” SHS Correlation ID
X-shs-contentid = “X-shs-contentid” “:” SHS Content ID
X-shs-localid = “X-shs-localid” “:” Local ID
X-shs-nodeid = “X-shs-nodeid” “:” Node ID
X-shs-arrivaldate = “X-shs-arrivaldate” “:” Successful
submission date and time
X-shs-duplicatemsg = “X-shs-duplicatemsg” “:” status
status = “yes” | “no”
```

The header `X-shs-duplicatemsg` support duplicate handling.

Examples of SHS extended header for return codes in asynchronous submission of messages.

Example:

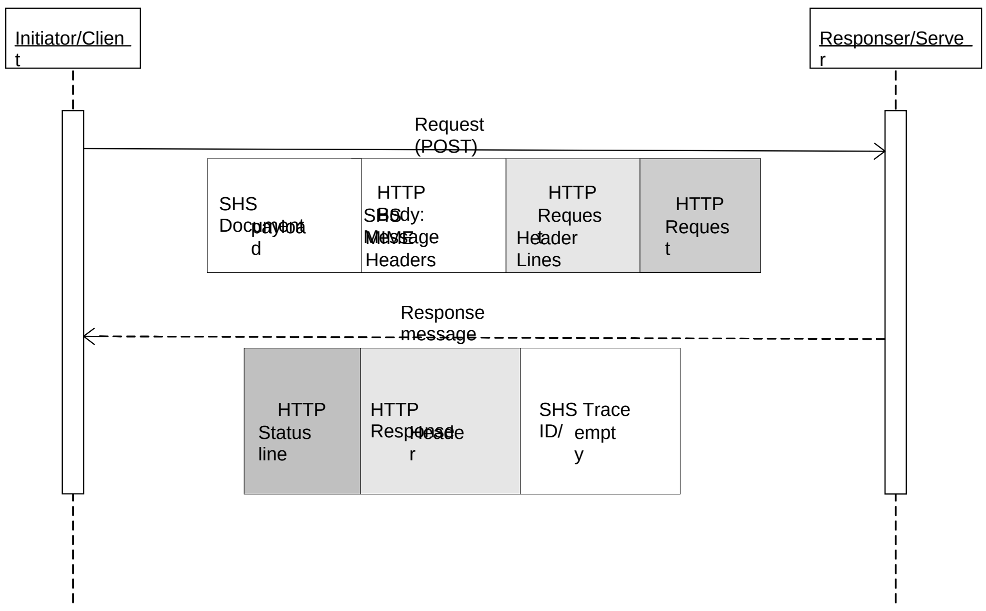
```
X-shs-txid: 11c00d35-078d-90a4-6042-80998091ca18
X-shs-corrid: RFV-VS-AKT-200405-111/2
X-shs-contentid: RFV-VS-AKT-200405-111
X-shs-arrivaldate: 2004-05-11T14:55:32
X-shs-duplicatemsg: no
```

I addition the response MAY contain the following data:

- SHS trace ID – when a request was successful from a client to an SHS server – applies to the Internal Message Access protocol.
- SHS error code and error information as text line
- There may be an empty message (headers only) – applies to SHS Backbone Communication protocol.

Example:

```
tx.id:11c00d35-078d-90a4-6042-80998091ca18
```



Asynchronous messages are delivered, using HTTP POST to the <RSURL>. The SHS message (multipart/mixed) is included in the message body.

Example:

```
HOST:  shs.rsv.se:xxxx
Content-length:  2198
Content.type:  message/rfc822

MIME-version:  1.0
Content-type:  multipart/mixed; boundary=----SHSMSG-17--
Subject: SHS message

... Here comes the actual SHS message ...
```

A normal HTTP response is returned, including status code, extended entity headers and a MIME answer. The status code is based on the standard codes used in HTTP. Some of the codes have an SHS-specific meaning, described below

- 202 Message is received
- 400 Illegal URL or illegal SHS Message
- 403 Not authorised to deliver messages to SHS
- 500 Internal server-error
- 503 SHS temporarily unavailable

The MIME formatted response data may be used by the receiving service to return the ID assigned by the SHS to the incoming message; otherwise an empty response is used.

Example:

```
202 Accepted
X-shs-txid: 11c00d35-078d-90a4-6042-80998091ca18
```

```
X-shs-corrid: RFV-VS-AKT-200405-071/2
X-shs-contentid: RFV-VS-AKT-200405-071
X-shs-localid: SHSID0123456789
X-shs-nodeid: NODE1
X-shs-arrivaldate: 2004-05-11T14:55:32
X-shs-duplicatemsg: no

MIME-Version: 1.0
Content-type: text/plain
SHSID0123456789
```

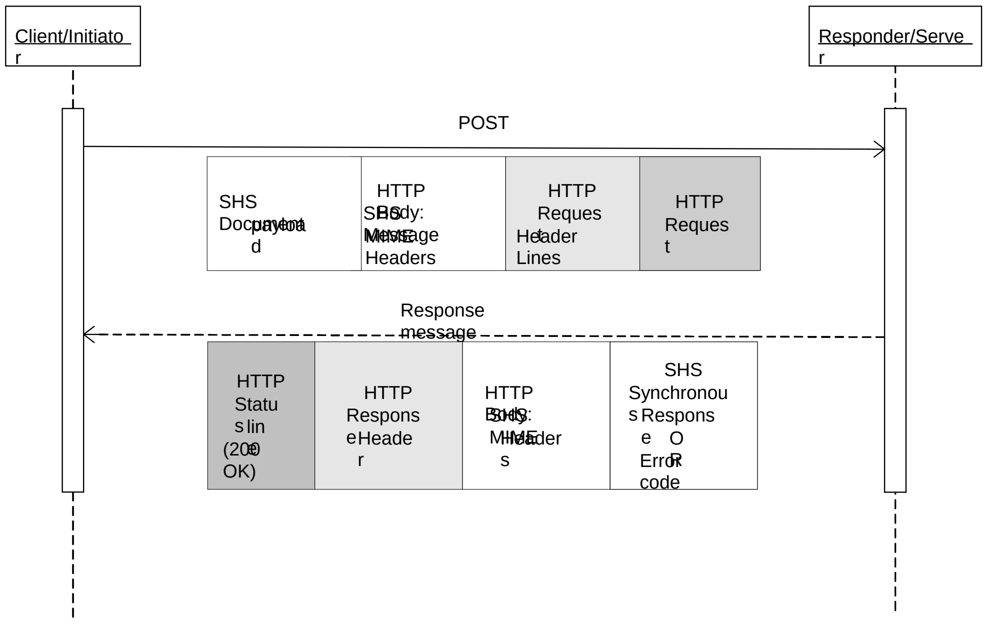
Error responses may include a clear text error message. This information is included in the message body as a simple text string followed by a new line.

Example:

```
400 Bad label

MIME-Version: 1.0
Content-type: text/plain
Illegal label
```

3.1.2 Synchronous variant (RS-sync)

Synchronous messages are managed similarly to asynchronous messages, except the content in the response message. The server MUST NOT return any response to the client before the remote system is responding or until a timeout has occurred. If response information is available it is returned in the HTTP response, otherwise an error code and informational text is returned.


Client/Initiator
Responder/Server
POST
SHS Document payload
HTTP Body: SHS Message MIME Headers
HTTP Request Header Lines
HTTP Request
Response message
HTTP Status line (200 OK)
HTTP Response Header
HTTP Body: SHS MIME Headers
SHS Synchronous Response OR Error code

The http return code is defined by the HTTP standard, but normally one of the following codes will be returned, including an SHS specific interpretation.

- 200 OK
- 204 The synchronous message has been sent and received, but no response message is available.
- 403 Not authorised to deliver messages to SHS
- 500 Internal server-error
- 503 SHS temporarily unavailable
- 504 Response timeout

When the return code 200 is used, the response message is included in the response. The response message is an SHS message included in the same way as in the sending phase (POST message).

Error codes SHOULD be returned as an extended entity header `X-shs-errorcode` with a clarifying error code. The error codes are defined in section 8.2. In addition error information SHOULD be returned in the returned content part for backward compatibility.

Definition:

```
X-shs-errorcode = ”X-shs-errorcode” ”:” Defined error code
```

Example:

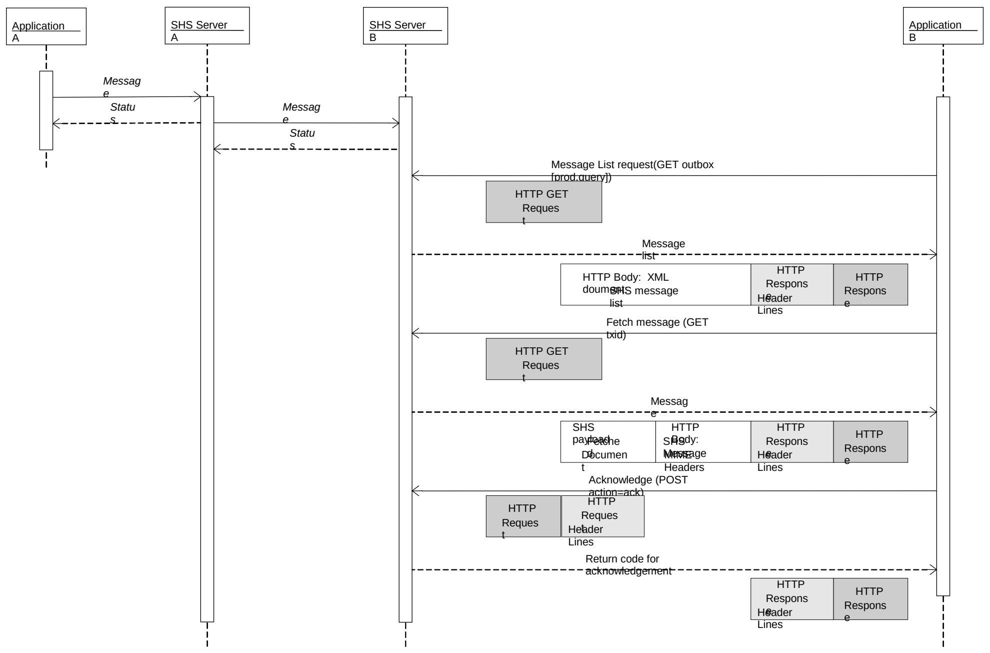
```
500 Internal Server Error
X-shs-errorcode: IllegalMessageStructure

MIME-Version: 1.0
Content-type: text/plain
Illegal message structure
```

3.2 Delivery service

The Delivery Service (DS) allows applications to connect to an SHS node and fetch asynchronously delivered SHS messages. SHS does not standardize the delivery of a synchronous message. This MAY be implemented by the SHS node as a plug-in that activates a specific application. This is further described in the Message Service chapter in the architecture document [ARCH].

3.2.1 Delivery Service (DS-async)

Asynchronous messages are fetched by business applications. This service is provided by the Delivery service which provides services for listing, fetching and acknowledgement of messages.


Application A
SHS Server A
SHS Server B
Application B
Message
Status
Message
Status
Message List request(GET outbox [prod;query])
HTTP GET Request
Message list
HTTP Body: XML doument SHS message list
HTTP Response Header Lines
HTTP Response
Fetch message (GET txid)
HTTP GET Request
Message
SHS payload Fetched Document
HTTP Body: SHS Message MIME Headers
HTTP Response Header Lines
HTTP Response
Acknowledge (POST action=ack)
HTTP Request
HTTP Request Header Lines
Return code for acknowledgement
HTTP Response Header Lines
HTTP Response

When fetching SHS-messages from an SHS node the fetching system contacts the SHS delivery service (DS) through HTTP/SSL. The delivery service has a URL <DSURL> working as a base for all communication.

HTTP GET URL governs the actual behaviour of the service. The response is then a formatted message.

If a transaction ID (txid) is specified DS returns the specific SHS message, otherwise it returns a list of available messages.

The following is a syntax description of the URL syntax. It is based on a BNF used in RFC2141.

```
<desturl>     ::= <messageurl> | <listurl>
<messageurl>  ::= <outbox> “/” [<txid>
<listurl>     ::= <outbox> “/” <product-type> [”?”
              <querystring> (“&”<querystring>)*]]
<outbox>      ::= <dsurl> “/” <shs-address>

<txid>        ::= <uuid>
<product-type>::= <shs-product>   # for SHS 1.1 compatibility
<querystring> ::= <filter> | <since> | <status> | <meta-attr>
              | <meta> | <originator> | <recipient> |
              <corrid> | <contentid> | <productype> |
              <sortattribute> | <sortorder> | <arrivalorder>
              | <maxhits>

filter        ::= “noack” (shows only non-acknowledged
messages)
since         ::= <time>
status        ::= “production” | “test”  default is
“production”
<meta-attr>   ::= “meta-“ <namepart> “=” <value> specifies
meta data to search for.
<meta>        ::= [“yes” | “no”] #include metadata, default
is “yes”.
```

```
<originator>  ::= originator ID
<endrecipient>::= end recipient ID
<corrid>      ::= correlation ID
<contentid>   ::= content ID
<producttype> ::= [<shs-product> [(“,”<shs-product>)*]]
<sortattribute> ::= “sortattribute=” <labelattr>
<sortorder>   ::= “sortorder=” <orderalt>
<arrivalorder> ::= “arrivalorder=” <orderalt>
<maxhits>     ::= maximum number of entries in list

<labelattr>   ::= “originator” | “from” | “endrecipient” |
“producttype” | “subject” | “contentid” | “corrid” |
“sequencetype” | “transfertype” | <metaname>
<orderalt>    ::= “ascending” | ”descending”  # Default is
              ”ascending”
<metaname>    ::= ”meta-” <namepart>
<namepart>    ::= string
<value>       ::= string

<dsurl>       ::= An URL to the delivery service

<shs-address> ::= Defined in SHS DTD Documentation [DTD]
<shs-product> ::= Defined in SHS DTD Documentation [DTD]
<uuid>        ::= Defined in SHS DTD Documentation [DTD]
<time>        ::= message according to ISO 8601 with the
format yyyy-mm-ddThh:mm:ss
```

As mentioned the two types of responses that the DS may provide is either a list of messages (directory listing) or an actual message. The following sections will describe the request and responses in more detail.

3.2.2 Message List

Message listing (directory listing) may be performed with or without a specified product type. When a specified product type is used the queryparameter producttype SHOULD be used, the specification of the product type in the <listurl> is for backward compatibility and is specified as:

`<listurl>  ::= <outbox> (/’ <product type>)`[2]`?`

In order to get a list of available messages an `HTTP GET` on the outbox URL is performed. The GET is combined with a query string, specifying the listing condition. If no condition is specified all messages will be listed. The following listing conditions are defined:

- `filter=noack` (shows only non-acknowledged messages)
- `since=[time]` message according to ISO 8601 with the format yyyy-mm-ddThh:mm:ss (shows only messages later <time>)
- `status=[production|test]`, default is ‘production’.
- `meta-<name>=<value>`, specifies meta data to search for.

[2] Use producttype list condition rather than product type in URL, which is for backward compatibility.

- `meta=[yes|no]`, include metadata, default is 'yes'.
- `originator=<originator ID>`
- `endrecipient=<endrecipient ID>`
- `corrid=<correlation ID>`
- `contentid=<content ID>`
- `maxhits=<maximum number of entries in list>`
- `producttype=<comma separated list of product types>`
- `sortattribute=<primary sort attribute>` see BNF in 3.2.1
- `sortorder=[ascending|descending]` Govens order for primary sort attribute if specified (sortattribute) - default is ascending
- `arrivalorder=[ascending|descending]` Order of secondary sort on submission time - default is ascending

If no primary sort attribute and sort order is specified the results SHALL be sorted on submission time. If arrivalorder isn't specified ascending submission order is assumed (first in – first out). If some or all messages in a listing do not contain an optional sort attribute value or meta attribute these messages SHALL be placed last in the resulting list.

Example: A listing without specified product type:

```
GET /ds/urn:X-shs:1234567890?filter=noack HTTP/1.1

GET /ds/urn:X-shs:1234567890?status=test&meta=no HTTP/1.1
```

A listing where the product type is specified (both examples with same result:

```
GET /ds/urn:X-shs:1234567890/urn:X-shs:07469eb8-078d-8c20-
789f-adab8091ca18?filter=noack HTTP/1.1

GET /ds/urn:X-shs:1234567890?producttype= urn:X-shs:07469eb8-
078d-8c20-789f-adab8091ca18 HTTP/1.1
```

A listing with multiple product types specified

```
GET /ds/urn:X-shs:1234567890?producttype= urn:X-shs:07469eb8-
078d-8c20-789f-adab8091ca18, urn:X-shs:dca43a5f-0389-f12d-
4548-8bc11a9b0d38 HTTP/1.1
```

Examples of request for a sorted listing

```
GET /ds/urn:X-shs:1234567890?sortattribute=originator&
filter=noack HTTP/1.1
GET /ds/urn:X-shs:1234567890?sortattribute=meta-
region&sortorder=descending&filter=noack HTTP/1.1
```

The list of available messages is returned in an XML document. Note that the returned content contains the XML structure only, no SHS message structures are returned for message lists.

Example:

```
<?xml version=”1.0”?>
<!DOCTYPE shs.message-list SYSTEM "shs-message-list-1.2.01.dtd">
<shs.message-list>
  <message tx.id=”a5268ffe-fc0b-11d2-802d-0060b0836211”
            timestamp=”2000-11-27T11:40:00”
            content.id=”RFV-VS-AKT-200011-071”
            corr.id=”RFV-VS-AKT-200011-071/2”
            size=”11723”
            from=”urn:X-shs:2021000548”
            to=”urn:X-shs:2021000985.Taxering”
            product=”urn:X-shs:a9268ffe-fc0b-11d2-802d-0060b0836299”
            sequence-type=”request”
            status=”production” >
    <meta name=”region”>gotland</meta>
    <meta name=”kategori”>tandl</meta>
    <subject>Taxeringsredovisning av tandläkare på Gotland för November
2000,
             från AKT/RFV</subject>
    <data datapartType=”TaxRedData”
          filename=”tax0011_09tl.trd”
          no-of-bytes=”10235”
          no-of-records=”189” />
    <data datapartType=”TaxRedSum”
          filename=”tax0011_09tl.trs”
          no-of-bytes=”148”
          no-of-records=”3” />
  </message>
  <message tx.id=”b5268ffe-fc0b-11d2-802d-0060b0836211”
            timestamp=”2000-11-26T23:17:00”
            content.id=”PRV-VS-KTN-200011-321”
            corr.id=”PRV-VS-KTN-200011-321/1”
            size=”52531”
            from=”urn:X-shs:2021000123”
            to=”urn:X-shs:2021000985.Företag”
            product=”urn:X-shs:b9268ffe-fc0b-11d2-802d-0060b0836299”
            sequence-type=”event” >
    <meta name=”period”>2000-11</meta>
    <meta name=”info”>nystartf</meta>
    <subject>Information om nystartade företag November 2000</subject>
    <data datapartType=”NystartFöretag”
          no-of-bytes=”51235” />
  </message>
</shs.message-list>
```

3.2.3 Fetching Messages

To fetch an individual message, a message URL is used. It is assembled by outbox and transaction ID of the message, separated by "/". Each request returns exactly one message.

Definition:

```
<messageURL>  ::= <outbox> “/” <txid>
<outbox>      ::= <dsurl> “/” <shs-address>
<txid>        ::= Message transaction ID
<dsurl>       ::= Delivery service URL
<shs-address> ::= Defined in SHS DTD Documentation [DTD]
```

The message is fetched with HTTP GET using the <messageURL>.

Example:

```
GET /ds/urn:X-shs:1234567890/urn:X-shs:11c00d35-078d-90a4-
6042-80998091ca18 HTTP/1.1
```

The response from the delivery service is a MIME formatted document containing the message.

Example:

```
HTTP 200 OK
MIME-version: 1.0
   … other HTTP headers …
   Content-type: message/rfc822
   Content-transfer-encoding: binary

   MIME-version: 1.0
   Subject: SHS-message
   Content-type: multipart/mixed; boundary=----SHSMSG-17--
   Content-transfer-encoding: binary
   ----SHSMSG-17--
   Content-Type: text/xml
   Content-Transfer-Encoding: binary
   <?xml version="1.0" encoding="iso-8859-1" ?>
   <!DOCTYPE shs.label SYSTEM "shs-label-1.1.dtd">
   <shs.label tx.id="beca5bd0-32e3-11d3-b8d6-0060b0836244" type="simple">
   ... shs label ...

   </shs.label>
   ----SHSMSG-17--
   Content-Type: application/octet-stream; name="<file name>"
   Content-Transfer-Encoding: binary
   Content-Disposition: attachment; filename=<filename>
   <?xml version="1.0" encoding="iso-8859-1" ?>
   <!DOCTYPE rsv.register SYSTEM "register-1.1.dtd">
   <rsv.register type="request">
   ... rsv register ...

   </rsv.register>
----SHSMSG-17—
```

```
Content-Type: application/octet-stream; name="<file name>"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=<file name>
MIAGCSqGSIb3DQEHAqCAMIIDpQIBATELMAkGBSsOAwIaBQAwgAYJKoZIhvcNAQcB
AACgggI3MIICMzCCAZygAwIBAgICAl4wDQYJKoZIhvcNAQEEBQAwPDELMAkGA1UE
BhMCU0UxETAPBgNVBAoUCFRlbGlhIENBMRowGAYDVQQDFBFUZWlsYSBUZXN0IFBL
----SHSMSG-17--
```

3.2.4 Acknowledge a fetched message

When an application has fetched an SHS message it may generate an acknowledgement. SHS will then send a delivery confirmation to the originator if requested.

The acknowledgement command is based on POST with an identical URL for a message used in the fetch operation, augmented by the query "action=ack"

Definition:

```
<ackURL> ::= <outbox> “/” <txid> “?action=ack”
<outbox>      ::= <dsurl> “/” <shs-address>
<txid>        ::= Message transaction ID
<dsurl>       ::= Delivery service URL
<shs-address> ::= Defined in SHS DTD Documentation [DTD]
```

Example

```
POST /ds/urn:X-shs:1234567890/urn:X-shs:11c00d35-078d-90a4-
6042-80998091ca18?action=ack HTTP/1.1
```

4 Communication patterns

While chapter 3 discusses the individual communication protocols, this chapter focuses on how end to end exchange is provided in SHS. The basic functionality of routing and error handling are visualised in sample communication patterns. The protocols will in themselves not limit the functionality to following patterns.

- Bilateral communication with response (request/reply)
- Bilateral communication without response (event).
- Failing variants of the above

4.1 Synchronous communication

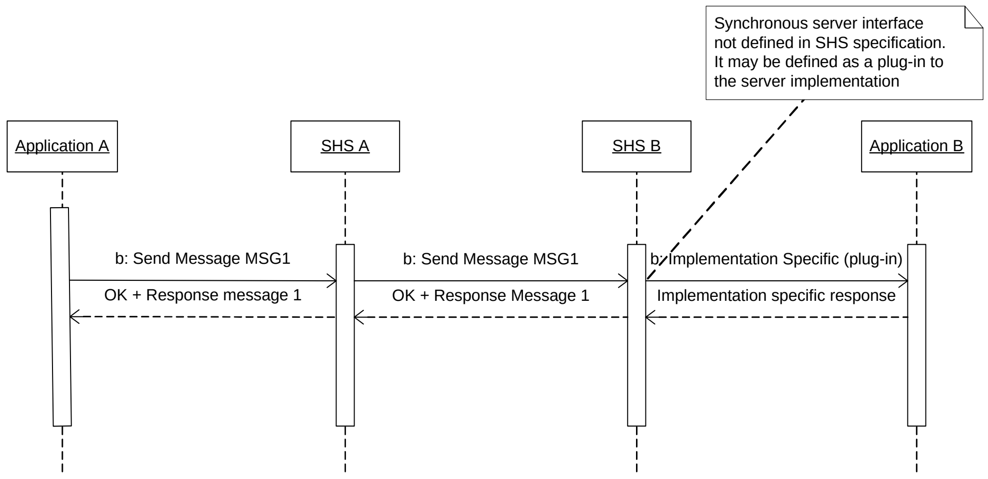
Synchronous communication is characterised by blocking of the calling application until either a result or an error message is returned. The Messaging Service builds and manages a chain of calls between applications. When a SHS node receive a request containing a SHS message with a transfer-type=synchronous it determines if the destination is handled locally or if the request must be routed to next SHS node.

The SHS node MUST return either an error code and error information as defined in 3.1.2 indicating an error condition, e.g. time-out situation or the response from next node in the chain.

The SHS transaction ID MUST be unique and used only once.

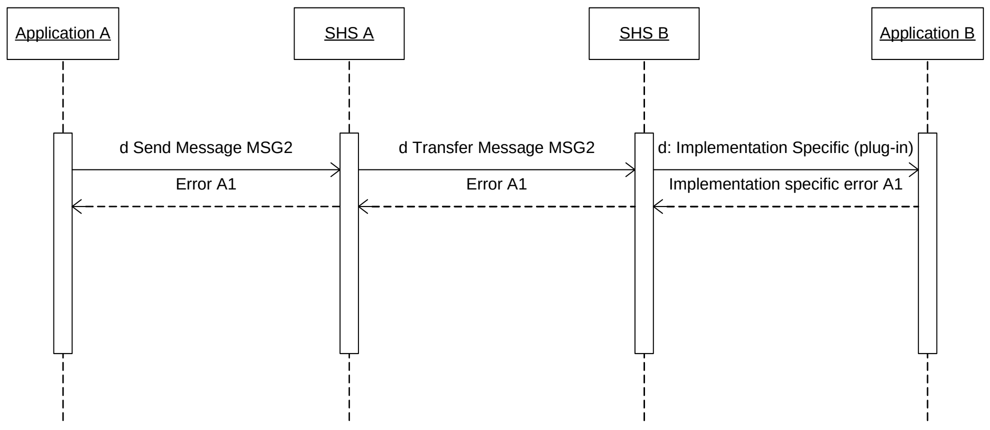
4.1.1 Bilateral communication with response

The following example describes the components and timing when a bilateral synchronous call is routed between applications via two SHS nodes.


Synchronous server interface not defined in SHS specification. It may be defined as a plug-in to the server implementation
Application A
SHS A
SHS B
Application B
b: Send Message MSG1
OK + Response message 1
b: Send Message MSG1
OK + Response Message 1
b: Implementation Specific (plug-in)
Implementation specific response

4.1.2 Bilateral communication that fails

When a synchronous call fails the error must be returned to the calling business application immediately



4.2 Asynchronous communication

Asynchronous communication is based on a store and forward mechanism that transfers the message one step at the time with intermediate storage, and therefore allows a reliable message transport even if all systems involved in the transfer is not reachable.

When an SHS node receives a request containing a SHS message with a transfer-type=asynchronous the receive service (RS) MUST store the message securely, before any response can be returned to the caller.

Errors that are detected directly may be signalled using the status codes described in 8.1.1, otherwise a separate SHS error message must be generated and sent back.

The SHS transaction ID MUST be unique and used only once. An SHS node MUST NOT resend an SHS message with a specific transaction ID if a response have been received for this given ID. An SHS node MUST make SHS messages with a given transaction ID available in the delivery service (DS) only once. An SHS delivery service (DS) MUST honour the acknowledgement of a message (POST action=ack).

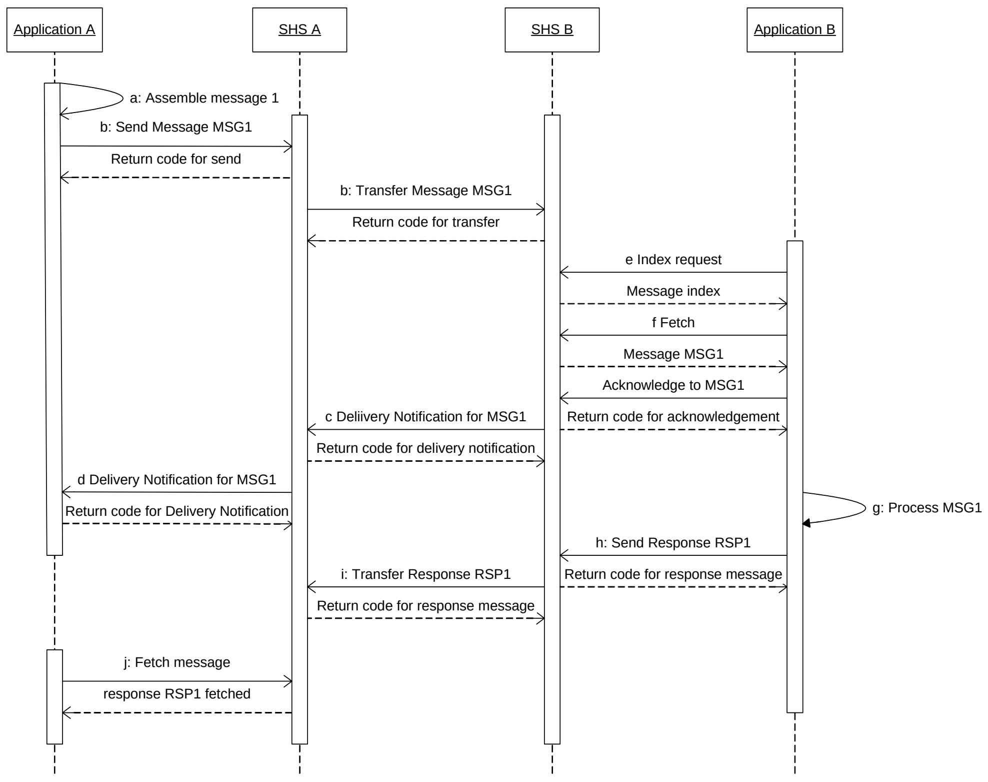
The following examples describe:

- The relationship between a single hop (link) and the end-to-end chain provided by the SHS messaging service.
- How errors may be immediately returned to caller (initiator)

- How errors may be returned as an error message at a later processing stage.

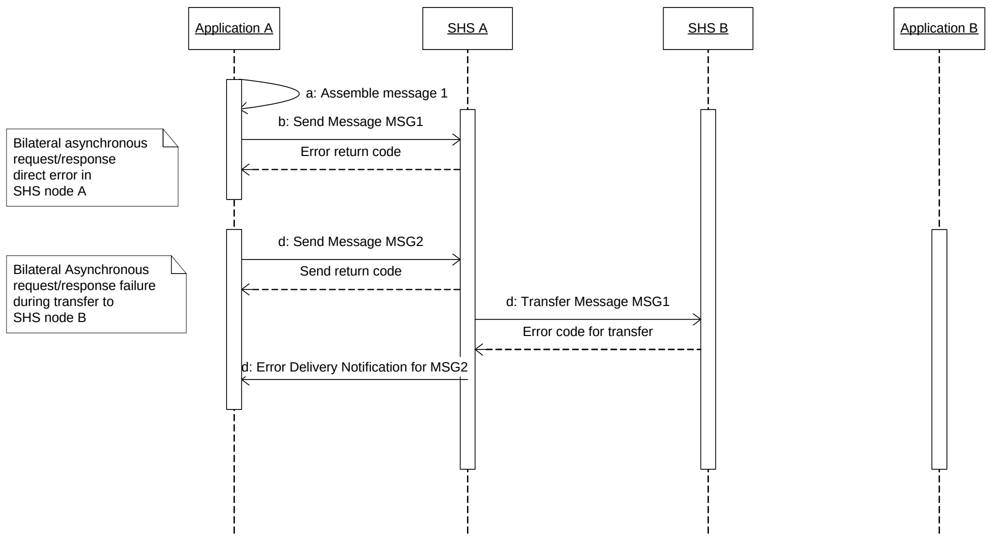
4.2.1 Bilateral communication with response

The vertical lines in the figure represent time. This indicates that since the communication is asynchronous only the individual calls (application – SHS, SHS – SHS etc) are dependent on the availability of other systems.



4.2.2 Bilateral communication that fails

Errors may be returned either directly or at a later stage as an error message. The following figure presents the two variants.


Application A
SHS A
SHS B
Application B
a: Assemble message 1
b: Send Message MSG1
Error return code
Bilateral asynchronous request/response direct error in SHS node A
d: Send Message MSG2
Send return code
Bilateral Asynchronous request/response failure during transfer to SHS node B
d: Transfer Message MSG1
Error code for transfer
d: Error Delivery Notification for MSG2

5 Transport layer protocols

5.1 General communication considerations

The transport layer may be considered as a usage profile of HTTP and HTTP over SSL/TLS (HTTPS), more specifically

- The POST method used by the initiating part (either application or SHS node) to send data to a “receive service” (RS) at next SHS node in the communication path and by receiving applications to acknowledge messages.

- The GET method used by the initiating application to fetch information (messages or a message listing) from a “delivery service” (DS) implemented at a SHS node.

- The use of persistent connections per default as defined in the RFC2616 chapter 8, or optionally using the keep-alive header for HTTP 1.0 backward compatibility as specified in chapter 19 of RFC2616.

- Error messages based on RFC2616 section 10, descriptions available in 8.1.

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 is 1024 bit for asymmetric keys and 128 bits for symmetric keys.

Implementations SHALL support SSL and SHOULD support TLS as specified in reference section page 3.

5.2 Port number conventions

Currently the following ports are used by SHS implementations. Installations are recommended to use these ports:

- SHS Backbone Communication: Port 8083 or 11288
- Internal Message Access: Port 8083 or 11292

6 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 business applications with the mechanism made available by the HTTP methods (the transport layer). In order to provide this the SHS layer is responsible for:

- Delivery using a store and forward mechanism (asynchronous mode).
- Immediate request and response (synchronous mode).
- Routing of SHS messages
- Exchange of agreements
- Configurable automatic delivery confirmation
- 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. These are defined in chapter 8.

6.1 Message format

The SHS messages SHALL conform to MIME and S/MIME as specified in [MIME] and [SMIME], with the following exceptions and clarifications.

- The MIME header Subject SHALL always be present. It SHALL contain either “SHS Message” (preferred) or “SHS document”.
- All types with text as main type, (i.e. text/xml), MUST represent new lines with CRLF.
- For the “Content-Type: text/xml”, a “name” attribute MAY be used.
- An optional “Content-Disposition” with “attachment” and “filename" MAY also be used for the label MIME part.
- Content-transfer-encoding. SHALL be defined as “binary” or “base64”.

Note that an entity in a multipart is not equal to a MIME-document according to the MIME standard. The difference is that a MIME-document must have a MIME version, while in the multipart part only fields starting with "Content-" have any meaning. In order to extract a part from a multipart message to a new MIME message, it is often necessary to change the header.

6.1.1 SHS message

An SHS message SHALL be based on the MIME 'multipart/mixed’ structure. The first part MUST be a text/xml part and contain the shs-label.

The following parts in the SHS message are data elements of MIME type as specified in the corresponding product type definition.

Example:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----SHS-MIMEPART-922348677--"
------SHS-MIMEPART-922348677--
Content-Type: text/xml
Content-Transfer-Encoding: binary
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE shs.label SYSTEM "shs-label-1.1.dtd">
<shs.label tx.id="beca5bd0-32e3-11d3-b8d6-0060b0836244" type="simple">
... shs label ...

</shs.label>
------SHS-MIMEPART-922348677--
Content-Type: application/octet-stream; name="<file name>"
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename=<filename>
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE rsv.register SYSTEM "register-1.1.dtd">
```

```
<rsv.register type="request">
... rsv register ...

</rsv.register>
------SHS-MIMEPART-922348677--
Content-Type: application/octet-stream; name="<file name>"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=<file name>
MIAGCSqGSIb3DQEHAqCAMIIDpQIBATELMAkGBSsOAwIaBQAwgAYJKoZIhvcNAQcB
AACgggI3MIICMzCCAZygAwIBAgICAl4wDQYJKoZIhvcNAQEEBQAwPDELMAkGA1UE
BhMCU0UxETAPBgNVBAoUCFRlbGlhIENBMRowGAYDVQQDFBFUZWlsYSBUZXN0IFBL
------SHS-MIMEPART-922348677----
```

6.1.2 Compound SHS-message

Compound messages is an extended service in SHS and MAY be supported in the SHS implementation. Support of compound messages MUST NOT be assumed.

A compound SHS message consists of other SHS messages. The Con-tent-type of these data parts is defined as ‘message/rfc822’.

Example:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----SHS-MIMEPART-922348677--"

------SHS-MIMEPART-922348677--
Content-Type: text/xml
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE shs.label SYSTEM "shs-label-1.1.dtd">
... label for the compound message ...

------SHS-MIMEPART-922348677--
Content-Type: message/rfc822
Content-Transfer-Encoding: binary

MIME-Version: 1.0
Subject: SHS-Message
Content-Type: multipart/mixed; boundary="----SHS-MIMEPART-922349432--"

------SHS-MIMEPART-922349432--
Content-Type: text/xml; name="<label name>.xml"
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE SHS.label SYSTEM "shs-label-1.1.dtd">
... label for first part ...

------SHS-MIMEPART-922349432--
Content-Type: application/octet-stream; name="<file name>"
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename=<filename>

... data file 1 in first SHS-message ...

------SHS-MIMEPART-922349432----
------SHS-MIMEPART-922348677----
```

6.2 Signing and encryption

SHS signing and encryption provides an end-to-end transmission security with respect to integrity and authenticity.

When SHS handles signed and/or encrypted data parts S/MIME version 2 [SMIME] SHALL be used. An SHS message MUST NOT be signed and encrypted in its entirety. The SHS labels MUST NOT be signed and encrypted.

Signed data SHALL use the header “Content-type: multipart/signed”.

Example SHS message with a signed data part:

Example:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----SHS-MIMEPART-922348677--"

------SHS-MIMEPART-922348677--
Content-Type: text/xml
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE shs.label SYSTEM "shs-label-1.1.dtd">
... cont. label ...

------SHS-MIMEPART-922348677--
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1; boundary="------SHS-MIMEPART-922348596--"

------SHS-MIMEPART-922348596--
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
name=smime.p7m
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename=smime.p7m

... data file ...

------SHS-MIMEPART-922348596--
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
```

```
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

------SHS-MIMEPART-922348596----

------SHS-MIMEPART-922348677----
```

SHS message with an encrypted and signed data part:

Example:

```
------SHS-MIMEPART-922348677--
Content-Type: text/xml
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE shs.label SYSTEM "shs-label-1.1.dtd">
... cont. label ...   --- lite av label.

------SHS-MIMEPART-922348677--
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1; boundary="------SHS-MIMEPART-922348596--"

------SHS-MIMEPART-922348596--
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
   name=smime.p7m
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename=smime.p7m

... encrypted, binary data ...

------SHS-MIMEPART-922348596--
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
```

```
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

------SHS-MIMEPART-922348596----

------SHS-MIMEPART-922348677----
```

6.3 Confirmation and error handling – asynchronous transfer

Asynchronous communication requires that status indications are handled as separate events or messages rather than return codes to calls. In SHS this is handled by messages of the special sequence type adm. There are three major types of status indications

- delivery confirmation that confirms that a specific message is successfully delivered to its final destination (SHS server)
- transport layer errors
- error messages that returns error conditions

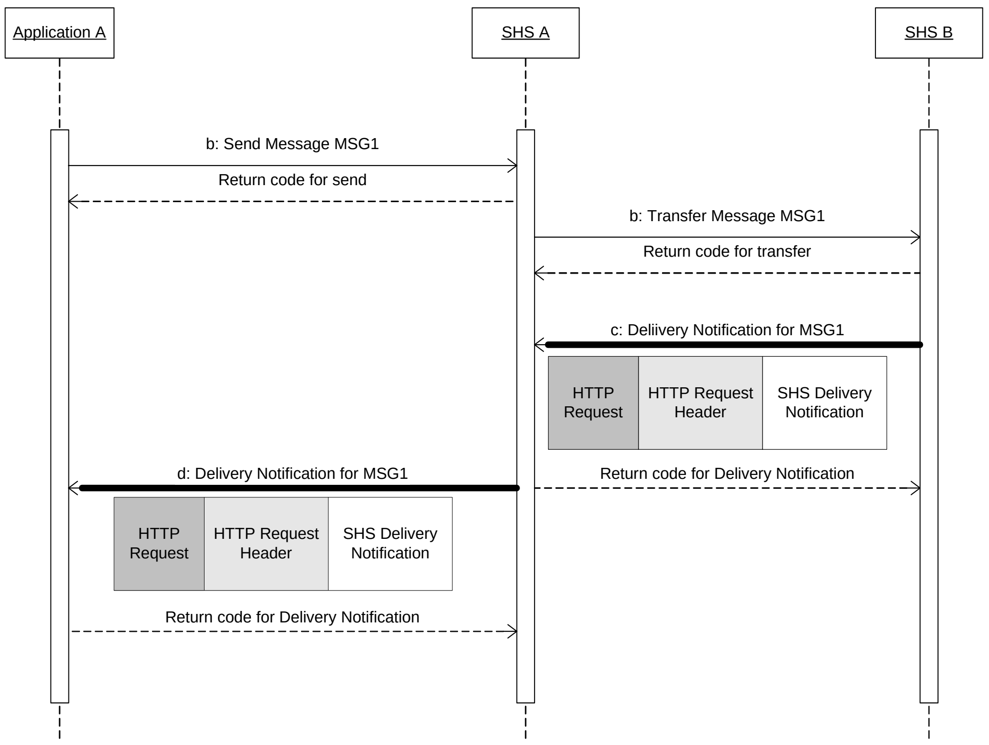
Both the confirmation and the error message must include the initial content id and correlation id.

6.3.1 Delivery Confirmation

Confirmation is governed by the “confirm” element in the agreement upon which a transmission is performed. By default no delivery confirmation is requested. If the delivery confirmation is requested the server MUST honour this request.

The SHS delivery confirmation message is defined by SHS management document DTD in [DTD]. The message is transferred by SHS in similar way as any asynchronous message. The sequence type SHALL be “adm”.

The information in the original SHS-message label is reused in the new confirmation message, with the content of the <from><originator> and <to><end-recipient> elements swapped. Content and Correlation ID is taken from original message.

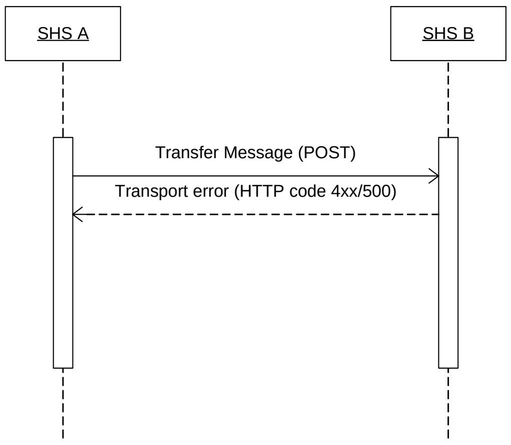

Application A
SHS A
SHS B
b: Send Message MSG1
Return code for send
b: Transfer Message MSG1
Return code for transfer
c: Deliivery Notification for MSG1
HTTP Request
HTTP Request Header
SHS Delivery Notification
d: Delivery Notification for MSG1
Return code for Delivery Notification
HTTP Request
HTTP Request Header
SHS Delivery Notification
Return code for Delivery Notification

Example:

```
<!DOCTYPE shs.management SYSTEM "shs-management-1.1.dtd">

<shs.management corr.id=”RFV-VS-AKT-199904-071/2”
                content.id=”RFV-VS-AKT-199904-071”>
  <datetime>1999-04-27T11:40:00</datetime>
  <confirmation/>
</shs.management>
```

6.3.2 Transport errors

The SHS layer protocols specifies transfer between neighbouring nodes (application to its SHS node and between SHS nodes) by interpreting the HTTP response codes so that they are useful in an SHS context. In the asynchronous communication error handling is handled by a separate error message, which also is handled by the SHS layer.



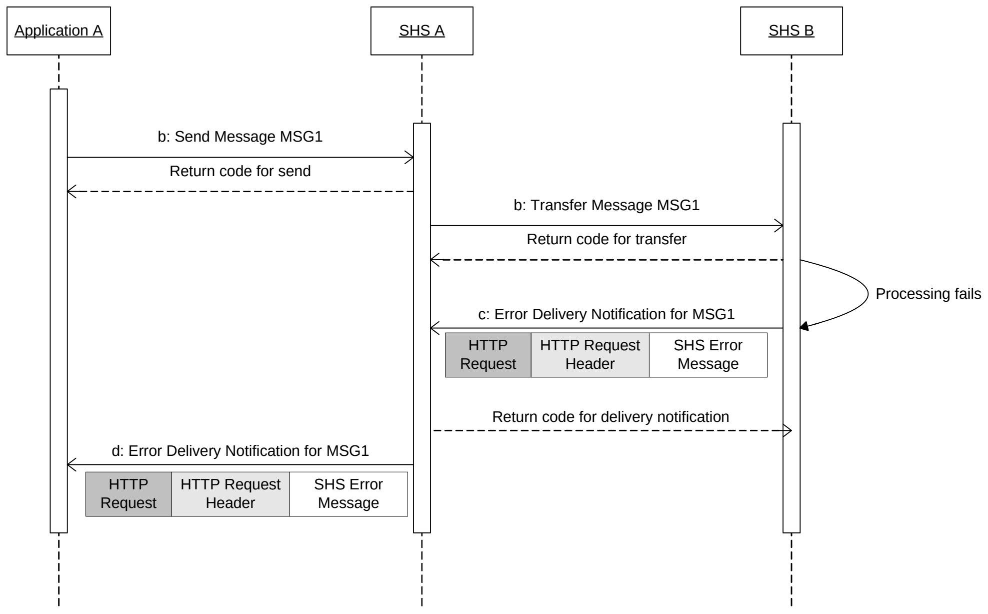
The error codes are specified in Transport error codes (8.1)

6.3.3 SHS Error messages

When an error situation occurs and the session to the sending part is closed or if the problem is not related to transport problems or occurs later in the chain of SHS nodes error conditions MUST be reported using the SHS error message. Error messages are sent using the SHS management document defined in [DTD].

The information in the original SHS-message label is reused in the new error message, with the content of the <from><originator> and <to><end-recipient> elements swapped. Content and Correlation ID is taken from original message.

The messages for SHS is defined in section 8.2 and SHOULD be used.



Example:

```
<?xml version=”1.0”?>
<!DOCTYPE shs.management SYSTEM "shs-management-1.1.dtd">

<shs.management corr.id=”RFV-VS-AKT-199904-071/2”
               content.id=”RFV-VS-AKT-199904-071”>
  <datetime>1999-04-27T11:40:00</datetime>
  <error errorcode=”UnknownReceiver”
         errorinfo=” Receiving actor unknown”>
    <appinfo name=”Receiver”>urn:X-shs:2021000000</appinfo>
  </error>
</shs.management>
```

6.4 Error handling - synchronous transfer

In the case of synchronous transfer the error status is returned immediately as a combination of:

- http status code (defined in 8.1.2)
- An SHS error code encoded in the extension header `X-shs-errorcode` defined in 3.1.2. Possible error codes are defined in 8.2.
- A clarifying error text (SHS error info described in 8.2) returned as data in the http body.

6.5 Routing

An SHS node routes a message according to information found in the SHS message label. If a receiver is stated in the label SHS uses this address, otherwise SHS looks for an agreement with correct product type and actors.

Using this information the SHS node can determine whether to deliver the message locally or to the RS of another SHS node.

For a more detailed description of routing actions, please see the routing section of [ARCH].

6.6 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 [Directory].

6.7 CA access protocol

An SHS implementation MUST validate certificates against the CA. Depending on CA either LDAP, OCSP or HTTP is used.

7 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.

Message specifications govern how SHS will handle a message. Examples of such information is

- the label attribute transfer-type that instructs SHS to handle a message as either synchronous or asynchronous
- the label attribute sequence-type that indicates the role of a message
- the product-type response required attribute, which governs if a receiver of a message is required to generate a response
- product information that declares if a particular information shall be encrypted and/or signed or not
- the label element meta-data that allow applications to create an annotation (intended for machine interpretation) to the content

7.1 Message flow

All information exchange between applications, including confirmation is sent as normal SHS messages with corresponding product type ID. All these messages are handled as standard SHS messages, which mean that SHS is only delivering the messages.

7.2 Confirmation and Error Messages

Errors may also occur after a successful delivery from a business system to SHS. This type of errors results in a new SHS-message containing an error message. Normally an illegal message or a transmission error causes these errors.

When a delivery confirmation or an error message is sent as an SHS message, it includes a label and one data part, which contains the message. In the label, some of the elements and attribute derive their contents directly from the original message.

The following arguments SHOULD be honoured by the business application:

The `corr.id` attribute of the `<shs.label>` element is the same as in the original message.

The `content.id` attribute of the `<shs.label>` element is the same as in the original message.

8 Error codes

8.1 Transport error codes

8.1.1 Asynchronous communication

- 202 Message is received
- 400 Illegal URL or illegal SHS-message (no message verification is required here, but if an error is detected here, the return code 400 should be used).
- 403 Not authorised to deliver messages to SHS (this should normally always be detected during the SSL negotiation). Note! It is NOT correct to use return code 401 (Unauthorized) when the sender doesn't have authority to send a message. This code is used for HTTP-authentication, which is not used by SHS.
- 500 Internal server-error
- 503 SHS temporarily unavailable

8.1.2 Syncronous communication

- 200 OK
- 204 The synchronous message has been sent and received, but no response message is available.
- 400 Message Handling Error (cause further described by error code and error information)
- 403 Not authorised to deliver messages to SHS (this should normally always be detected during the SSL negotiation). Note! It is NOT correct to use return code 401 (Unauthorized) when the sender doesn't have authority to send a message. This code is used for HTTP-authentication, which is not used by SHS.
- 500 Internal server-error
- 503 SHS temporarily unavailable
- 504 Response timeout

8.2 Error messages

The SHS error message is defined in the SHS management message [DTD] and is used to return error status for asynchronous transfers. Two fields are used.

- errorcode – a mnemonic representation of the error. Allowed values are defined below
- errorinfo – supplementary information. The values SHOULD be configurable. Values for Swedish and English implementation is provided below.

In a synchronous transfer the codes and information SHOULD be used:

- errorcode in the http extended header X-shs-errorcode
- errorinfo as a text line in the http content (plain text – not encoded as SHS Message)

Table 2 SHS Error codes

Error Code	**Error Information (SV)**	**Error Information (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
IllegalOriginator	Otillåtet värde på skapare	Illegal syntax for originator
IllegalEndRecipient	Otillåtet värde på slutmottagare	Illegal syntax for end recipient
IllegalMessageStructure	Otillåten struktur på SHS-meddelandet	Illegal Message Structure
IllegalDatapartContent	Ej validerbart innehåll i en eller flera datadelar	One or more dataparts contains illegally formatted information
OtherError	Annat oväntat fel (*bör beskrivas inom parenteser*)	Ohter error (clarified within parenteses)