SHS Version 1.0.01 Internal Web Services Interface (IWSI)

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 provide information about how to use this Web Service Interface as an SHS interface in a business application.

1.1 Audience

This document is primarily intended for the developers of the business applications.

1.2 References

[BASE64] RFC2045, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (BASE64)

[SOAP] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[WSDL] Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[DTD] SHS Version 1.2.01 DTD Description

[Protocols] SHS Version 1.2.01 Protocol

1.3 Document history

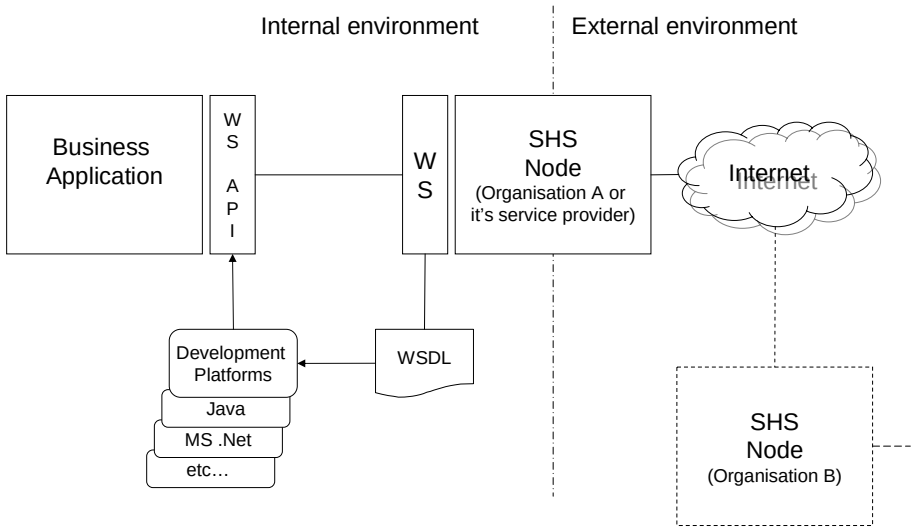
Version	Date	Change	By	Approved
0.1	2003-04-30	Initial structure	John Olsén	
0.2	2003-05-27	Updates from Luleå workshop	John Olsén	
0.9	2003-09-24	Updates after ÄR 2003-09-02	Christer Marklund, Anders Lindgren	
1.0	2003-10-09	Final version (included into Programming Interfaces)	Anders Lindgren	Jan Lundh
1.0.01-A	2004-05-19	First draft (A) of update of the Internal Web Services Interface version 1.0.01 • Changes from SHS 1.2.01 • Clarification of error handling	Anders Lindgren	
1.0.01-B	2004-05-26	Second draft (B) of updated IWSI. Comments from SHS ÄR 040526	Anders Lindgren	
1.0.01-C	2004-06-04	Third draft moved to separate document	Anders Lindgren	
1.0.01-D	2005-02-01	Comments from ÄR 041123	Anders Lindgren	
1.0.01	2007-04-05	Final subversion	Anders Lindgren	Christer Marklund Jan Lundh

2 Overview

The Internal Web Services Interface version 1.0 is compatible with SHS 1.2. The reasons behind this interface are mainly to provide a complement to the existing interfaces that:

- require a minimum of installed SHS specific components or libraries
- benefit from the tools/development platforms that are developed for the web services technology
- is platform independent

This interface is named SHS Internal Web Services Interface since it provides an interface for business applications to access SHS services. These services include (but is not limited to) reliable and secure transport of information and audit logging. SHS provides the connection to other organisations or applications, defined as “external environment” in the figure below.



SHS Internal Web Services Interface will not use UDDI directory service for publication of the WSDL file. Instead the definition file (WSDL file) will be available for download from a regular web server.

3 Restrictions and Versions

3.1 Restrictions

SHS Internal Web Services Interface version 1.0 is restricted to the following functions:

- sending messages (synchronously and asynchronously)
- retrieving of message lists (list of messages available) and messages

Version 1.0 does not support:

- signing and encryption on SHS message level
- compound messages

The size of the business documents that may be handled in this version will be limited to 1 MB (One Megabyte).

Note! When encoding of the message is base64, the actual data transferred is 1.33 times larger than the payload. This means that the actual data that must be handled is approximately 1.3 MB rather than 1 MB.

3.2 SOAP version

The SHS Internal Web Service Interface 1.0 is based on WSDL version 1.1 [WSDL] and SOAP version 1.1 [SOAP].

Reasons for SOAP 1.1 are primarily vendor interoperability and a broad availability of development platforms. The tentative versions of the WS-I Basic Profile have also influenced the design.

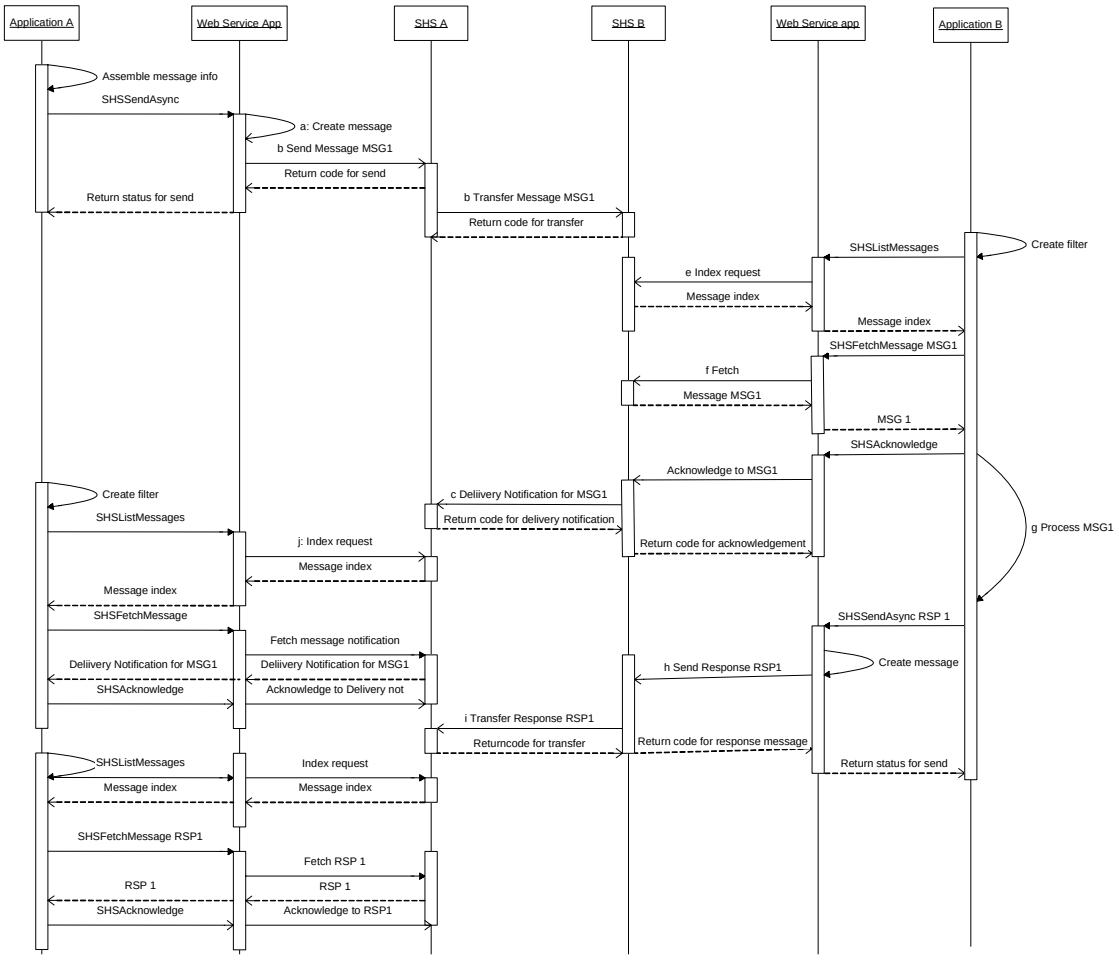
SOAP 1.2 has been considered as it includes features (i.e. MIME-part attachments) that would solve some of the technical difficulties that come with handling of opaque data in XML. At the time of design of the SHS Internal Web Services Interface SOAP version 1.2 has not yet reached the status of recommendation from W3C. It is assumed that there will be limited availability of interoperable, generally available development platforms during the initial lifetime of this specification.

The drawback of the SOAP 1.1 approach is that it requires XML syntactically correct information as payload. The interface therefore requires base64-encoding of the information payload. This means that business documents MUST be BASE64 encoded by the business system before submitted to the Web Service.

4 SHS – Web Service Interface

All documents that are sent through the Web Service Functions have to be BASE64 encoded. The server side decodes the documents before submitting them to the SHS node. When retrieving documents through the Web Service the documents are BASE64 encoded by the server and must be decoded by the client.

The communication pattern “bilateral with response” performed with use of the Internal Web Service Interface is illustrated by the following diagram over the logical sequence[1]. The delivery notification used here is optional and may be used when the sending application need immediate notification while the actual processing of data occurs later e.g. in a daily batch.



[1] The logical or abstract sequence – due to the environment the web service server and SHS server may actually initiate next hop before the return status is sent. In the diagram return status is always sent before next operation for simplicity.

4.1 Data structures (complex types)

These data structures are used in functions both for sending and receiving. Complex types are used foremost due to limitations found in handling more than one return parameter and to keep data together in lists.

Several parameters are optional. Support for optional parameters may depend on the actual platform used for the Web Service server. In this description as well as in the enclosed WSDL an optional parameter may be either omitted (minoccurs=0) or provided as empty string (nillable).

4.1.1 cMessage

This main structure defines the actual SHS message and is used for both sending (SHSSendAsync, SHSSendSync) and fetching (SHSFetchMessage). Note that at least one of the parameters *From* and *Originator* must be provided.

Complex type	cMessage		
Parameter name	**Data type**	**Occurs**	**Comment**
From	xsd:string	[0..1]	States the actor that sends the message. Optional value. Provide an empty string if it should be omitted.
To	xsd:string	[0..1]	The receiving actor. Optional value. Provide an empty string if it should be omitted.
Originator	xsd:string	[0..1]	End user that created the document. Optional value. Provide an empty string if it should be omitted.
EndRecipient	xsd:string	[0..1]	Final recipient of the message. Optional value in the DTD, provide an empty string if it should be omitted.
ProductTypeID	xsd:string	[1..1]	Product type
Meta	list of cMeta	[0..*]	Optional value. Provide empty list if it should be omitted. see cMeta section X
TxID	*xsd:string*	*[0..1]*	*Optional parameter for use when document transfers are reinitiated*
ContentID	xsd:string	[0..1]	Defined by the sender. Identifies the information in the data. Optional value. Provide an empty string if it should be omitted*. If ContentID is left blank it must be generated by the web service server*.
Subject	xsd:string	[0..1]	Textual description of content. *Optional.*

CorrelationID	xsd:string	[0..1]	Defined by the sender, used to associate sent and received messages. Optional value. Provide an empty string if it should be omitted. If corr.ID is left blank it must be generated by the web service server.
SequenceType	*xsd:string*	*[1..1]*	*Must have one of the following values "event" | "request" | "reply" | "adm"*
Status	xsd:string	[0..1]	P|T Production or test, if left blank then production is assumed
AttachedDataParts	list of cAttachedDataPart	[0..*]	See section X, cAttachedDataPart

Complex type	cMessage		
Parameter name	**Data type**	**Occurs**	**Comment**
From	xsd:string	[0..1]	States the actor that sends the message. Optional value.
To	xsd:string	[0..1]	The receiving actor. Optional value.
Originator	xsd:string	[0..1]	End user that created the message. Optional value.
EndRecipient	xsd:string	[0..1]	Final recipient of the message. Optional value.
ProductTypeID	xsd:string	[1..1]	Product type
Meta	list of cMeta	[0..*]	Optional value. Provide empty list if it should be omitted. see cMeta section 4.1.7 below.
TxID	xsd:string	[0..1]	Optional parameter for use when SHS message transfers are reinitiated
ContentID	xsd:string	[0..1]	Defined by the sender. Identifies the information in the data. Optional value. If ContentID is omitted it will be generated by the web service server.
Subject	xsd:string	[0..1]	Textual description of content. Optional.
SequenceType	xsd:string	[1..1]	States the role played by this message in an information exchange. event|request|reply|adm. Mandatory in SHSSendAsync – ignored by SHSSendSync.
CorrelationID	xsd:string	[0..1]	Defined by the sender, used to associate sent and received messages. Optional value. If corr.ID is left omitted it will be generated by the web service server.
Status	xsd:string	[0..1]	P|T Production or test Production is assumed if omitted

AttachedDataParts	list of cAttachedDataPart	[0..*]	See section 4.1.2, cAttachedDataPart, below

4.1.2 cAttachedDatapart

This structure contains a business document and meta data about the document.

Complex type	cAttachedDatapart		
Parameter name	**Data type**	**Occurs**	**Comment**
DataPartInfo	cDataPartInfo	[1..1]	Information about the datapart (business document/payload)
EncDocument	xsd:base64Binary	[1..1]	The base64 encoded business document.

4.1.3 cDataPartInfo

This structure contains meta data about data parts transferred in the message.

Complex type	cDataPartInfo		
Parameter name	**Data type**	**Occurs**	**Comment**
DataPartType	xsd:string	[1..1]	Specifies the type of data part. Should be a copy of the data part type attribute from the corresponding description in the product type definition
Filename	xsd:string	[0..1]	The name of the file attached to the message. Optional value.
NoOfBytes	xsd:unsignedInt	[0..1]	Number of bytes for this business document. Optional value. Provide zero (0) if it should be omitted.
NoOfRecords	xsd:unsignedInt	[0..1]	Number of records for this business document. Optional value. Provide zero (0) if it should be omitted.

4.1.4 cListedMessage

Parameters associated with a message in a message listing.

Complex type	cListedMessage		
Parameter name	**Data type**	**Occurs**	**Comment**
TxID	xsd:string	[1..1]	The transaction ID of the message.
Timestamp	xsd:date	[1..1]	Indicates when the message became ready to fetch. Date according to ISO 8601 extended format 'yyyy-mm-ddThh:mm:ss'.
CorrelationID	xsd:string	[0..1]	The correlation ID of the message.
ContentID	xsd:string	[0..1]	The content ID of the message
Size	xsd:unsignedInt	[0..1]	Total number of bytes of the message.
Originator	xsd:string	[0..1]	End user that created the message.
From	xsd:string	[0..1]	States the actor that sends the message.
To	xsd:string	[0..1]	The receiving actor.
EndRecipient	xsd:string	[0..1]	Final recipient of the message.
ProductTypeID	xsd:string	[0..1]	Product type.
Subject	xsd:string	[0..1]	Subject. Informational, free form text.
SequenceType	xsd:string	[1..1]	States the role played by this message in an information exchange. event|request|reply|adm
Status	xsd:string	[1..1]	P|T Test or production. Default "production"
Data	list of cDataPartInfo	[0..*]	See section 4.1.3, above
Meta	list of cMeta	[0..*]	see section 4.1.7 cMeta, below

4.1.5 cFilter

This structure contains the parameters that may be used to filter message listings, see [DTD] for further details.

Complex type	cFilter		
Parameter name	**Data type**	**Occurs**	**Comment**
NoAck	xsd:string	[1..1]	Specifies whether acknowledged messages should be included in the returned message list or not. T|F
Since	xsd:date	[0..1]	Sets a lower bound date for messages returned in

			the list. Optional value.
Status	xsd:string	[0..1]	Productions or test. Decides if test or production messages should be returned. (P|T) Production is default if parameter is omitted
Meta	list of cMeta	[0..*]	Optional value. Provide empty list if it should be omitted.
Originator	xsd:string	[0..1]	Select messages from a specific originator. Optional value
EndRecipient	xsd:string	[0..1]	Select messages for a specific end recipient. Optional value
CorrelationID	xsd:string	[0..1]	Select messages with a specific correlation ID. Optional value
ContentID	xsd:string	[0..1]	Select messages with a specific content ID. Optional value
ProductTypeID		[0..*]	Selects messages for specific product types. Multiple values are allowed. If omitted all messages of all product types are listed.
MaxHits	xsd:unsignedInt	[0..1]	Upper limit for returned number of messages.

4.1.6 cSort

This structure contains parameters that governs the sort order of a message list and is an optional in parameter to SHSListMessages function.

Complex type	cSort		
Parameter name	**Data type**	**Occurs**	**Comment**
SortAttribute	xsd:string	[0..1]	Optional attribute for primary sort order of resulting message list. Datetime is default sort attribute May have the values: “originator” | ”from” | “endrecipient” | “producttype” | “contentid” | “corrid” | ”sequencetype” | ”transfertype” |”meta-*”
SortOrder	xsd:string	[0..1]	Sort order for attribute specified in SortAttribute – may have the values ascending|descending – default is ascending
ArrivalOrder	xsd:string	[0..1]	Secondary sort on submission time – may have the values ascending|descending default is ascending

4.1.7 cMeta

This structure contains user defined meta data about the message. This data is mainly intended for automatic computer interpretation, see [DTD] for further details.

Complex type	cMeta		
Parameter name	**Data type**	**Occurs**	**Comment**
Name	xsd:string	[1..1]	Name of the meta attribute e.g. “region”, “dnr”
Value	xsd:string	[1..1]	The value of the meta attribute

4.1.8 cSHSSendAsyncRet

This structure is used for returned data from the *SHSSendAsync* function.

Complex type	cSHSSendAsyncRet		
Parameter name	**Data type**	**Occurs**	**Comment**
TxID	xsd:string	[1..1]	UUID for message transaction
CorrID	xsd:string	[1..1]	Defined by the sender, used to associate sent and received messages.
ContentID	xsd:string	[1..1]	Defined by the sender
LocalID	xsd:string	[1..1]	Local ID generated by SHS node
NodeID	xsd:string	[1..1]	SHS Node identity
ArrivalDate	xsd:string	[1..1]	First successful submission time. If this time is in the past this submission is a duplicate message for the receive service.
DuplicateMsg	xsd:string	[1..1]	Set to “yes” if SHS node detects a duplicate message, otherwise “no”

4.2 Functions for sending documents

There are two functions for sending documents, one asynchronous and one synchronous. Both functions provide the Web Service server with all the information needed to create and send an SHS message.

The list of in-parameters only differs in that the asynchronous function has the parameter *SequenceType* that indicates the role played by the message in an information exchange.

Note that at least one of the parameters *From* and *Originator* must be provided in both SHSSendAsync and SHSSendSync.

When providing parameters with URN syntax to the Web Service (*From, To, Originator, EndRecepient* and *ProductTypeID*) the prefix "urn:X-shs:" should be omitted.

4.2.1 SHSSendAsync

This function sends the documents and the information that the web service server need in order to create a label. The web service server thereafter creates an SHS message, including label and attached documents, and submits it to an SHS node for processing. Note that the client must BASE64 encode the documents before using them as parameters.

Function name	SHSSendAsync		
Parameter name	**Value**	**Data type**	**Comment**
In parameters			
Message	See section 4.1.1, cMessage	cMessage	
Out parameters			
cSHSSendAsyncRet	See section 4.1.8, cSendAsyncRet	cSendAsyncRet	Returns status codes defined in the cSendAsyncRet complex type. See [Protocols] for a description of possible status codes.
Fault	General SOAP faults and SHS server faults that generate error situations are signalled as SOAP fault. Further explained in section 5 below.		

4.2.2 SHSSendSync

This function has the same functionality as SHSSendAsync except that it receives a message in return. The initializing SHS message must have a sequence-type of "request" – this is always set by the web service server. The documents returned are BASE64 encoded and must be decoded by the client.

Function name	SHSSendSync		
Parameter name	**Value**	**Data type**	**Comment**

In parameters		
Message	See section 4.1.1, cMessage cMessage	
Out parameters		
MessageReturn	See section 4.1.1, cMessage cMessage	
Fault	General SOAP faults and SHS server faults that generate error situations are signalled as SOAP fault. Further explained in section 5 below.	

4.3 Functions for retrieving documents

The Web Service Interface consists of two functions for retrieving data, one for retrieving a list of messages (a list of unique transaction ID's for messages) and one for retrieving the messages one by one using the messages transaction ID in the received list.

4.3.1 SHSListMessages

This function returns a message list that refers to SHS messages found in the out-box of the SHS node. A filter may be provided to limit the list returned. The message list is always sorted on timestamp (see complex type cListedMessage). Of the parameters returned only "TxID" is required by the SHSFetchMessage function to retrieve actual messages.

Function name	SHSListMessages		
Parameter name	**Value**	**Data type**	**Comment**
In parameters			
Filter	See section 4.1.5, cFilter cFilter		Used to filter lists, the filter uses meta-fields in the label.
To	SHS-address	xsd:string	Receivers address (URN without URN prefix – normally organisation number) , optionally with internal addressing. Example: "2021000985" or "2021000985.internaladdr"
SortParameters	See section 4.1.6	cSort	Optional parameters for sort order – if omitted messages are returned in same order as submitted (ascending datetime)
Out parameters			
ListedMessage	See section **Fel! Hittar inte referenskälla.**,	list of cListedMessage	List of messages sorted on timestamp parameter available in the cListedMessage

	cListedMessage	complex type.
Fault	General SOAP faults and SHS server faults that generate error situations are signalled as SOAP fault. Further explained in section 5 below.	

4.3.2 SHSFetchMessage

This function returns the document(s) included in the SHS message with indicated transaction ID (TxID). The documents are BASE64 encoded by the Web Service server upon reception and must be decoded by the client.

Function name	SHSFetchMessage		
Parameter name	**Value**	**Data type**	**Comment**
In parameters			
TxID	transaction.ID	xsd:string	Transaction ID in UUID format
To	SHS-address	xsd:string	Receivers address (URN without URN prefix – normally organisation number) , optionally with internal addressing. Example: “2021000985” or “2021000985.internaladdr”
Out parameters			
MessageReturn		cMessage	
Fault	General SOAP faults and SHS server faults that generate error situations are signalled as SOAP fault. Further explained in section 5 below.		

4.3.3 SHSAcknowledge

This function is used to acknowledge a successful reception of an SHS message.

Function name	SHSAcknowledge		
Parameter name	**Value**	**Data type**	**Comment**
In parameters			
TxID	transaction.ID	xsd:string	
To	SHS-address	xsd:string	URN for receivers address, and optionally internal address.
Out parameters			

Fault	General SOAP faults and SHS server faults that generate error situations are signalled as SOAP fault. Further explained in section 5 below.

5 Fault handling

The SOAP standard defined a method to signal error and status conditions named "SOAP Faults"[2]. The following error conditions are returned as SOAP Fault:

- General SOAP Faults – faults generated by the SOAP routines and SOAP server itself.
- SHS server faults that generates http error status.

Errors that occur when processing an asynchronous message is returned in a separate SHS error message. The status information returned by the SHS server when an asynchronous message is submitted is returned in a separate complex type "cSHSSendAsyncRet" rather than as SOAP fault.

The SOAP errorcode MAY contain the standard values defined in SOAP 1.1 [SOAP]. For SHS specific errors the values "server" or "client" MUST be used as errorcode. In addition the SHS errorcode (X-shs-errorcode) defined in [Protocols] SHOULD be included in the SOAP Fault faulttext element.

SOAP Faultcode "server", which basically matches http status "5xx", is used when the error is generated due to malfunction or faulty configuration on server side, i.e. the client may retry the call later without modification. On the other hand the client application MUST make changes before retries when the faultcode "client" is encountered. Errounous SHS message syntax or denied access is examples of errors where the faultcode is set to "client". These correspond to the http transport errors 400 and 403.

[2] SOAP Fault for SOAP 1.1 is defined in "Simple Object Access Protocol (SOAP) 1.1 section 4.4"

6 Enclosures

SHSIWSIv101.wsdl: WSDL template file