SARS External Technical Interface Specification

ANDRE RABIE
EXECUTIVE: TECH. ARCHITECT
-2019683
3 NOV 2017
Signature:

Approve ARB 30/10/2017

DOCUMENT INFORMATION

Version Number	**1.0.5**
Version Date	**2017-10-26**
Compiled by	**Nic Kakoulla**
Security Classification	**Restricted**

TABLE of Contents

1 Document Control

1.1 Revision Control

The document has being through the following revisions.

Name	Date	Change	Version
Nic Kakoulla	2017-06-01	Created	1.0.0
Nic Kakoulla	2017-07-18	Updated to include: JMS, AMQP and Security	1.0.1
Nic Kakoulla	2017-08-30	Updated Communication Protocols	1.0.2
Nic Kakoulla	2017-09-18	Updated Protocols to show EDI preferred Communication Protocol	1.0.3
Nic Kakoulla	2017-09-28	Updated Communication Protocols and Glossary	1.0.4
Nic Kakoulla	2017-10-26	Included a note in the communication Protocols	1.0.5

Table 1: Revision Control

1.2 Reference Documents

Name	Description	Date	Version
SARS ICC Error Handling Functional Specification.	Documents the error handling strategy that need to be applied by all applications integrating with the SARS.	2014-11-17	0.8

Table 2: Reference Documents

2 Glossary

Acronym	Definition
API	Application Programming Interface
EDI	Electronic Data Interchnage
ESB	Enterprise Service Bus
FTE	WebSphere MQ File Transfer Edition
GUID	Global Unique Identifier
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol with SSL
ID	Identifier
JMS	Java Message Service
MQ	Message Queueing
QM	Queue Manager
QoS	Quality Of Service
REST	RESTFul Web Services
SACU	Southern African Customs Union
SARS	South African Revenue Services
SFTP	Secure File Transfer Protocol
SiBus	Service Integration Bus
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
WSDL	Web Services Definition Language
XML	Extensible Mark-up Language

Table 3: Glossary

3 Terminologies

Term	Definition
Service Consumer	Requesting System
Service Provider	System providing Service
Consumer Request Queue	This is the queue where the consumer will send a request to the Service Enabler which in turn will send a request to the Service Provider
Consumer Response Queue	The queue from which the consumer retrieves any response messages
Provider Request Queue	The queue on which the Provider listens and retrieves any requests messages from the Service Enabler
Provider Response Queues	The queue on which the Provider responds to the Service Enabler for each request message received
Queue Manager	IBM MQ system service that provides a logical container for the message queue, and is responsible for transferring data to other queue managers via message channels
Intermediary	An intermediary is a program/application that interjects some function between two end-user programs.
Application	A Sub-system consisting of technologies and components owned by the same technical Team

Table 4: Terminologies

4 Introduction

4.1 Purpose

The purpose of this document is to define the External Technical Interface Specification for integrating applications across SARS and other external entities following Service Oriented Architecture (SOA) as a standard approach.

The interface specification described in this document will allow for the introduction of applications with minimum effort, by leveraging off a shared communication infrastructure and common Message Format. The adoption of industry standards and leading practices further realises this goal.

4.2 Scope

The scope of this document is to provide an interface specification through which any external application can interact with any service requestor/provider application in the SARS integration environment in a reliable manner.

This interface specification, once implemented by an application, will allow an application to become an extension of the SARS central architecture.

This is achieved through specifying the Integration pattern, Communication Protocol and Message Format Standards & Rules to be adhered to when integrating in the SARS environment.

4.3 Exclusions

This document excludes general security considerations. It is assumed that any application that connects to SARS has been authenticated and authorised by the SARS security infrastructure.

Additionally, each service provider will be responsible for creating and maintaining documentation describing their service set.

5 Communication Protocols

The following protocols will be adopted for Integration, depending on the technology on which the application is built:

SARS Preferred Messaging and Web Services Protocols in order of preference:

1	WebSphere MQ
2	SOAP Web Services, Java Messaging Service (JMS)
3	RESTful Web Services
4	Advanced Message Queuing Protocol (AMQP)

Table 5: SARS Preferred Messaging and Web Services Protocols in order of preference

SARS Preferred File Transfer Protocols in order of preference:

1	Connect:Direct
2	Sterling File Gateway (SFG)
3	WebSphere MQ File Transfer Edition (MQFTE)
4	Secure File Transfer Protocol (SFTP)

Table 6: SARS Preferred File Transfer Protocols in order of preference

SARS EDI Gateway Preferred Protocols in order of preference:

1	AS2
2	AS3
3	X.400

Table 7: SARS Preferred File Transfer Protocols in order of preference

Note:

- X.400 will be discontinued end of October 2018;
- SFTP will only be maintained for external companies currently interfacing with SARS. Will be decommissioned in the future.

5.1 WebSphere MQ

5.1.1 Overview:

The primary Communication Protocol Standard for SARS is WebSphere MQ. Whilst proprietary in nature, it has become the industry standard queueing technology owing to its robust nature and support for various platforms.

5.1.2 Key Characteristics:

The following are some of the key characteristics of MQ:

- Asynchronous, Store-and-Forward type processing;
- Guaranteed Delivery;
- Performance and Ability to handle large volumes; and
- Platform and Language Independence.

5.1.3 Usage and Exclusions:

The following are some considerations regarding the usage of MQ:

- MQ is the preferred protocol for transactional type integration and should be used wherever possible.
- Other protocols should only be considered in the following instances:

    - Provider is a packaged application; or
    - Integration with external organisations that do not have the MQ infrastructure/required skills.

5.1.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in a role of a service consumer using the MQ protocol:

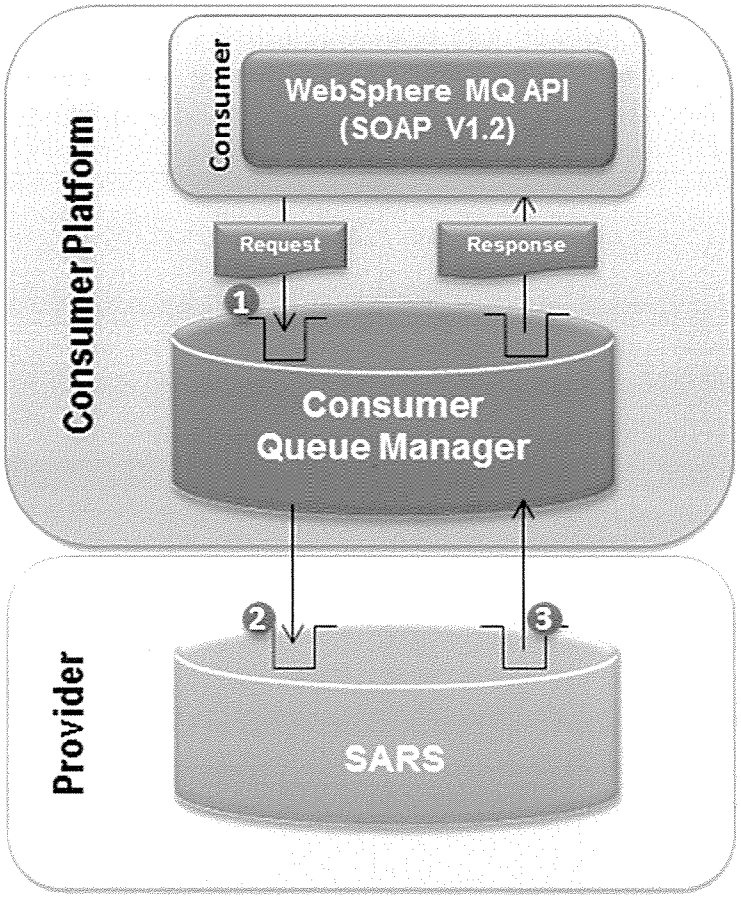


Figure 1: Typical interaction between an Application and SARS using the MQ protocol

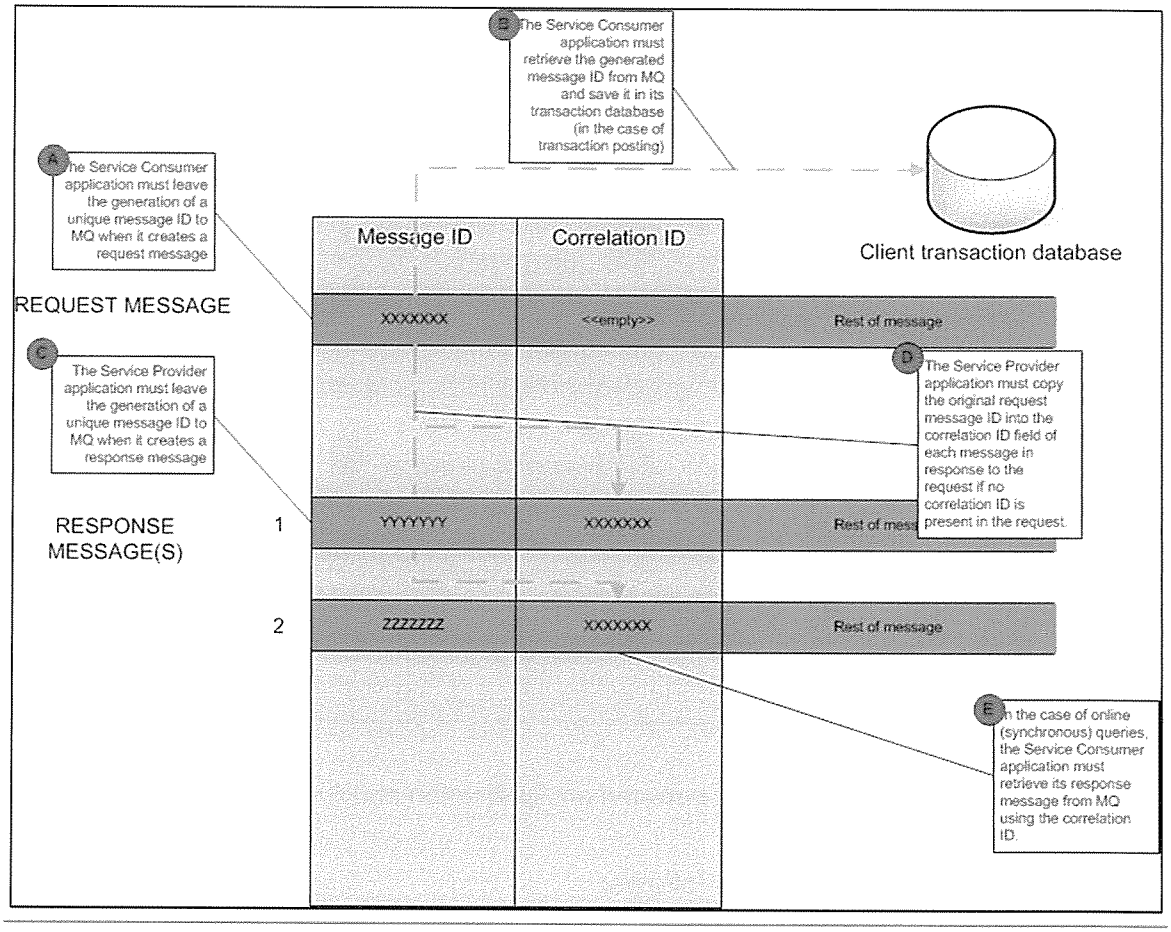
The steps involved are as follows:

1. The consumer builds a request message and puts it on the designated request queue of its Queue Manager via the MQ API.
2. The MQ infrastructure moves the message to the corresponding request queue on the SARS Queue Manager.
3. SARS sends the response message to the consumer's Queue Manager.

NB: The interaction when the application acts a provider/destination follows the same steps but in reverse order (i.e. read from request queue and put on response queue).

5.1.5 Correlation:

Due to the Asynchronous nature of MQ, Standards and Rules are necessary for correlating/relating a response message to the associated request message. WebSphere MQ provides message header fields for a message ID and correlation ID. These fields are used to tie request/response message pairs together. By default, MQ auto-generates a unique message ID when a message is created. This is the behaviour that all message creators must follow (both Service Consumer and Service Provider applications).



Rule 1 : *The Service Consumer must allow MQ to generate the message ID.*

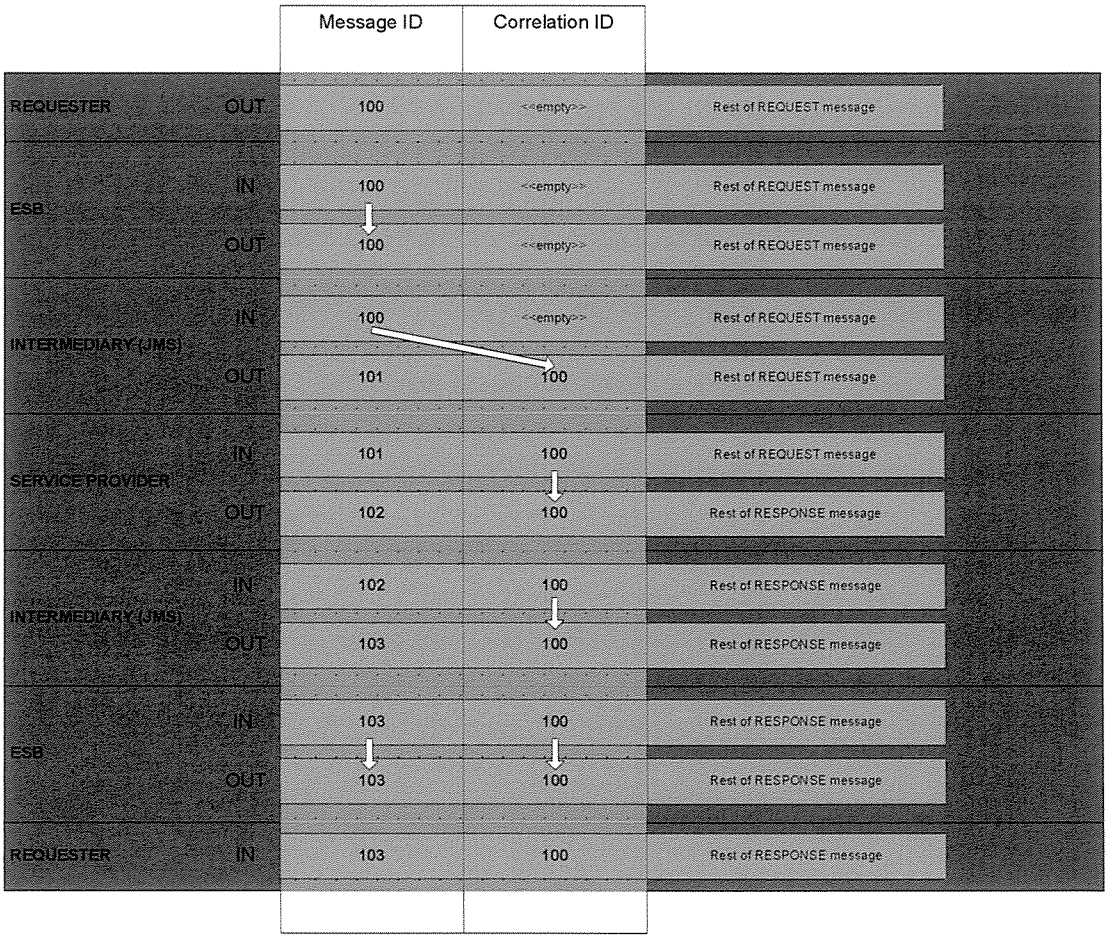
The Service Consumer application will be responsible for retrieving the message ID from MQ and storing the message ID in a database for message tracking (in the case of transaction posts) or for synchronous retrieval of the response message by correlation ID (queries). This leads to rule 2 below.

Rule 2: *The Service Consumer must retrieve the generated message ID and use it for response correlation.*

It may happen that the request message passes through one or more Intermediaries (other than SARS) and that these Intermediaries are unable to

preserve the message ID of the request message. In this case, the components must behave in the following manner: if a request message it receives contains a populated correlation ID field, this field must be preserved when it creates a new message; otherwise, the component must copy the message ID from the incoming message to the correlation ID of the outgoing message.

The diagram below depicts such a scenario:



Rule 3: *Any message processing component must ensure the correlation ID field of the MQ message header is preserved if it is present on the incoming message.*

Rule 4: *If any message processing component (Intermediary) is unable to preserve the MQ message ID, it must copy the incoming message ID into the correlation ID field of the outgoing message. If the correlation ID is already present, Rule 3 applies.*

Rule 3*: The Service Provider must first apply rule 3 then in the absence of a pre-existing correlation ID in the request message, it must copy the request message MQ message ID into the correlation ID of the response message MQ header.*

5.1.6 Naming Convention:

The naming convention below applies to both request and response queues and will be supplied by SARS in the Logical Integration Design (LID) document.

Format: *<Channel Prefix>.<Service name>.<Queue type>*

- **<Channel Prefix>:** Application's Channel ID
- **. Dot separator**
- **<Service name>:** This name describes the function(s) to be performed by the service.
- **. Dot separator**
- **<Queue type>:** Specified as REQ for a request message and RES for a response message

E.g. EXT.SAMPLESERVICE.REQ
EXT.SAMPLESERVICE.RES

5.1.7 Quality of Service (QoS) Setup:

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The MQ QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Asynchronous
Delivery Assurance	Guaranteed Delivery
Availability	24 / 7
Expiration Time	7 Days
Timeout	60s (Front-end only)
Transactional (Y / N)	Y
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Preferred Size	Less than 10MB
Throughput	1000 per second

Table 8: Configurable MQ QoS Parameters

5.1.8 Error Handling:

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.1.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.

5.2 Web services

5.2.1 SOAP – Web Service

5.2.1.1 Overview

Web services describes a standardised way of integrating Web-based applications using Extensible Mark-up Language (XML) , Simple Object Access Protocol (SOAP), Web Services Definition Language (WSDL) and open standards over an Internet protocol backbone (HTTPs). XML is used to tag the data, SOAP is used to structure and transfer the data and WSDL is used for describing the services.

5.2.1.2 Key Characteristics

- Platform and language independent;
- Requires both applications to be available during message calls;
- No additional technology required to use and in format which packaged services are provided; and
- Synchronous and Asynchronous processing.

5.2.1.3 Usage and Exclusions:

Below is the Web specification version that needs to be adhered to when an application integrates with SARS via Web Service:

- SOAP 1.2;
- WSDL 2.0; and
- WS-I 1.2

The following are some considerations regarding the usage of the Web Service protocol:

- Web services are the industry standard for integration. They are limited in terms of usage to the following scenarios:
  - Integration with packaged applications (Packaged Applications usually expose their functionality through a Web Service); and
  - Integration with external organisations where the service being integrated already exist as a Web Service or the organisation cannot invest in MQ.

5.2.1.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in using the Web service protocol:

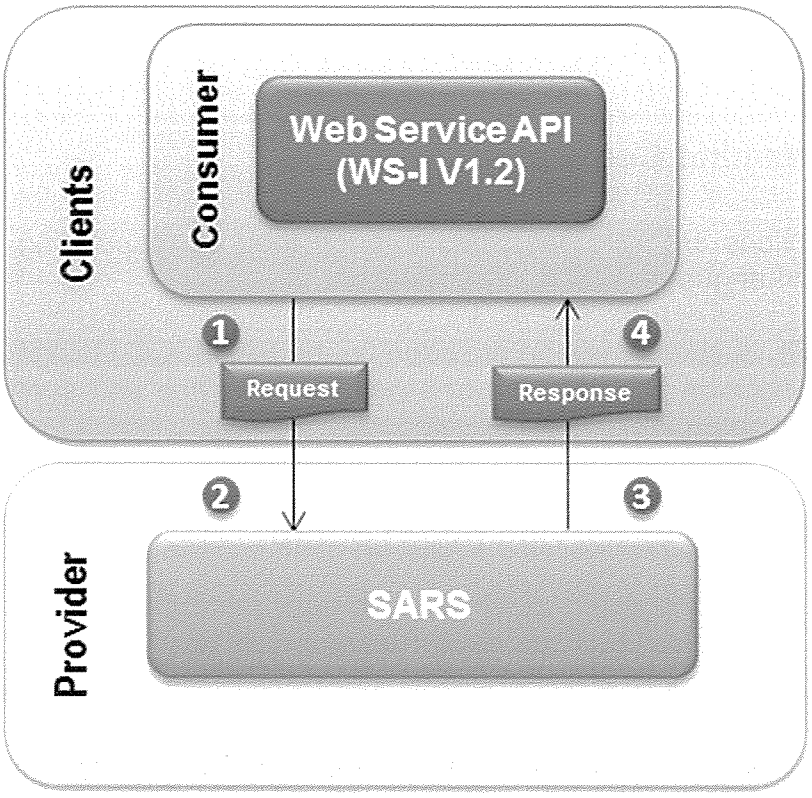


Figure 2: Typical interaction between Consumer and Provider using the Web service protocol

The steps involved are as follows:

1. The consumer builds a request message and sends it to SARS via its Web Service API.
2. The SOAP input node receives the message and invokes the corresponding process.
3. A response is sent to the consumer.
4. This Web Service call is concluded and the response returned to the consumer.

NB: The interaction when the application acts a provider/destination follows the same steps but in reverse order (i.e. exposes a Web service exposed via SARS).

5.2.1.5 Correlation

Web services calls are synchronous in nature and as such there is no specific correlation between the request and the response message. If the message call spans multiple web services, the header must be used to correlate the response to the request.

5.2.1.6 Naming Convention

Web Services are fully described through their corresponding Web Service Definition Language (WSDL) specification.

The WSDL document will be provided by the SARS ICC Design Team as part of the Integration Specification of a particular Service and configured as part of the associated Integration Development.

5.2.1.7 Quality of Service (QoS) Setup

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The Web service QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Synchronous
Delivery Assurance	N/A
Availability	24 / 7
Expiration Time	60s
Timeout	60s
Transactional (Y / N)	N
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Throughput	100 per second

Table 9: Configurable web service QoS Parameters

5.2.1.8 Error Handling

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.2.1.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.
- SARS will not manipulate data.

5.2.2 RESTful Web Services

5.2.2.1 Overview

Representational state transfer (REST) or RESTful web services are one way of providing interoperability between computer systems on the Internet. REST-compliant web services allow requesting systems to access and manipulate textual representations of web resources using a uniform and predefined set of stateless operations.

5.2.2.2 Key Characteristics

- Platform-independent;
- Language-independent;;
- Standards-based (runs on top of HTTPs); and
- Can easily be used in the presence of firewalls.

5.2.2.3 Usage and Exclusions:

Below is the Web specification version that needs to be adhered to when an application integrates with SARS via RESTful Web Service:

- HTTP 1.1;;
- CSV/PSV;
- XML;
- JSON; and
- RSS

The following are some considerations regarding the usage of the REST protocol:

- REST relies on a stateless client-server, cacheable communications protocol. In most cases, HTTPs protocol is used.
  - Data is uniquely referenced by URL and can be acted upon using HTTPs operations (GET, PUT, POST, DELETE, etc.…)
  - Integration with external organisations where the service being integrated uses HTTPs methods as an organisation cannot invest in MQ.

5.2.2.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in using the RESTful Web service protocol:

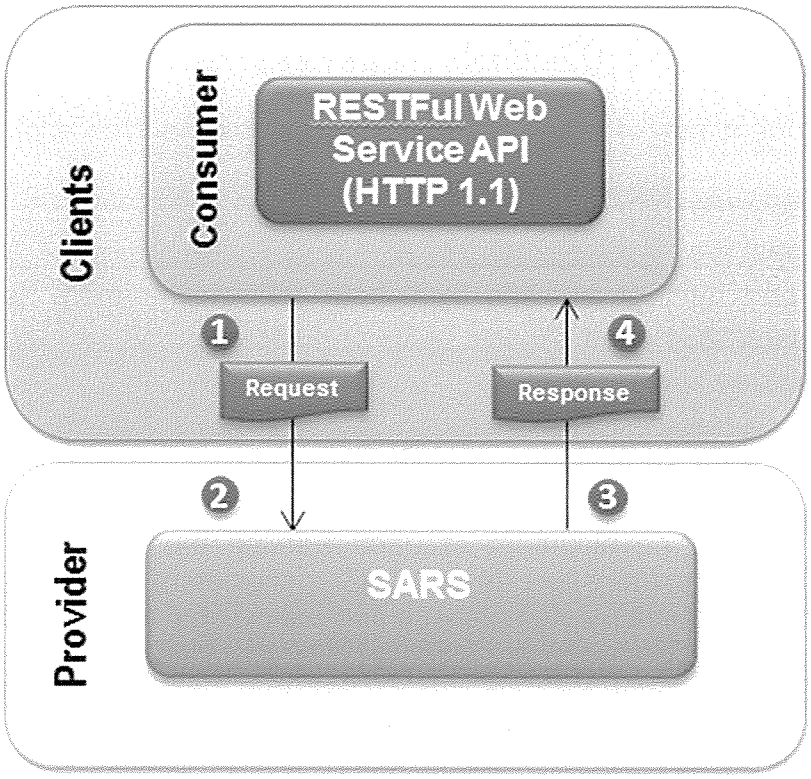


Figure 3: Typical interaction between Consumer and Provider using the RESTful Web service protocol

The steps involved are as follows:

1. The consumer builds a request message and sends it to SARS via its RESTful Web Service API.
2. The input node receives the message and invokes the corresponding process.
3. A response is sent to the consumer.
4. This Restful Web Service call is concluded and the response returned to the consumer.

NB: The interaction when the application acts a provider/destination follows the same steps but in reverse order (i.e. exposes a RESTful Web service exposed via SARS).

5.2.2.5 Correlation

RESTful Web services calls support synchronous and asynchronous calls and as such there is no specific correlation between the request and the response message. If the message call spans multiple web services, the header must be used to correlate the response to the request.

5.2.2.6 Naming Convention

URIs are identifiers of resources that work across the Web and consist of the following:

- A scheme (http or https);
- A host (i.e./ www.example.org);
- A port number followed by a path with one or more segments (i.e./ /users/1234); and
- A Query string.

5.2.2.7 Quality of Service (QoS) Setup

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The RESTful Web service QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Synchronous Asynchronous
Delivery Assurance	N/A
Availability	24 / 7
Expiration Time	N/A
Timeout	N/A
Transactional (Y / N)	N
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Throughput	100 per second

Table 10: Configurable RESTful web service QoS Parameters

5.2.2.8 Error Handling

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.2.2.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.
- SARS will not manipulate data.

5.3 Connect:Direct

5.3.1 Overview

Connect:Direct provides a secure and managed file transfer capability across a wide range of platforms and networks within and between organisations.

5.3.2 Key Characteristics

- Guaranteed Delivery;
- Reliable handling of large-sized files with the ability to resume;
- Platform and Language Independence; and
- Uses a secured connection to transfer files.

5.3.3 Usage and Exclusions:

The following are some considerations regarding the usage of Connect:Direct (C:D):

- Connect:Direct is the preferred protocol for external Batch type integration and should only be deviated from if the external organisation(s) cannot invest in Connect:Direct Infrastructure.

5.3.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in a role of a transfer source using the Connect:Direct (C:D) protocol:

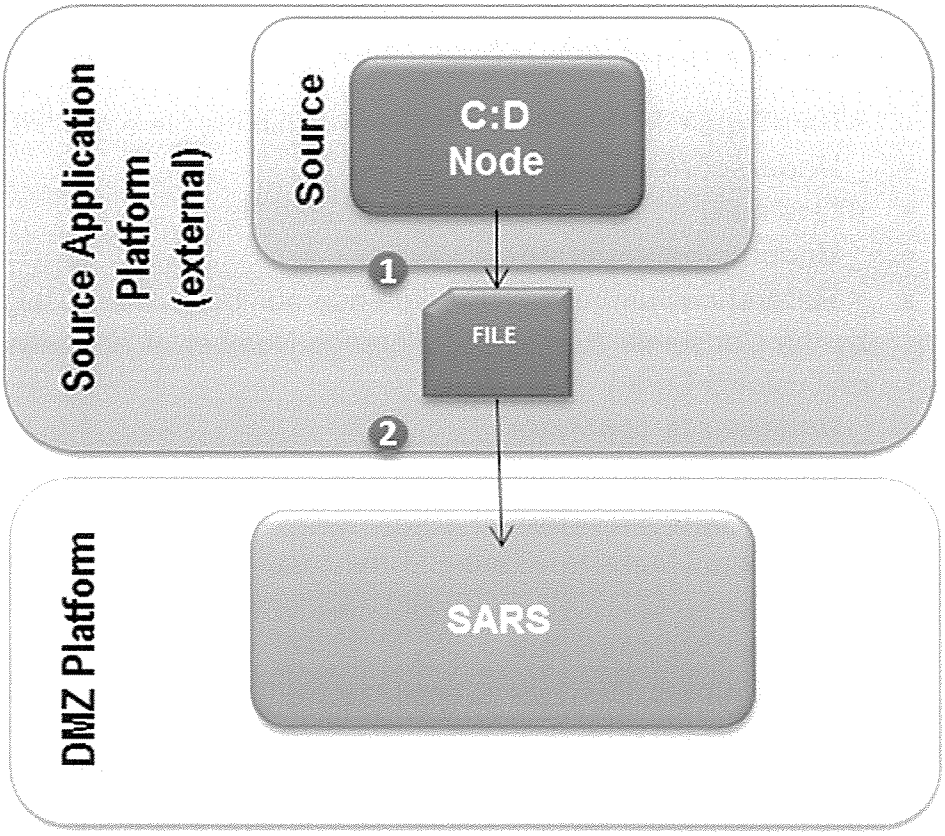


Figure 4: Typical interaction between an Application and SARS using Connect:Direct (C:D) protocol

The steps involved are as follows:

1. The external file transfer source generates the file and places it locally.
2. The File transfer source executes the file transfer script and sends the file to SARS via the De-militarised Zone (DMZ).

NB: The interaction when the application acts as a destination follows the same steps but in reverse order (i.e. Connect:Direct Node receiving the file from the DMZ)

5.3.5 Correlation:

There is no specific correlation requirements defined for the Connect:Direct protocol and as such, it should be handled as part of the file content.

5.3.6 Naming Convention:

The file naming convention below applies to both request and response files:

5.3.6.1 The Request File

Format: <DataType>_<VersionNr>_<ReferenceNumber>_<Unique File ID>_<MessageCreateDateTime>_<Node name>

- **< DataType>**:Type of data contained within the file e.g. WHD
- **_ Underscore separator**
- **<VersionNr >:** The version number of the file.
- **_ Underscore separator**
- **< ReferenceNumber>:** The reference number will be assigned by the Consumer (E.g. Tax Reference Number of an individual or Customs Code)
- **_ Underscore separator**
- **< Unique File ID >:** Unique ID assigned to the file.
- **_ Underscore separator**
- **< MessageCreateDateTime>:** Timestamp in which the file was created using CCYYMMDDTHHMMSS format.
- **_ Underscore separator**
- **<Node name>:** The Source application's C:D node name.

E.g. WHD_2_8880000008_67890_20111025T160508_ITSolutions.txt

5.3.6.2 The Response File

Format: <DataType>_<VersionNr>_<ReferenceNumber>_<Unique File ID>_<MessageCreateDateTime>_ < File Response Code>_<Node name>

- **< DataType>**:Type of data contained within the file e.g. WHD
- **_ Underscore separator**
- **<VersionNr >:** The version number of the file.
- **_ Underscore separator**
- **< ReferenceNumber>:** The reference number will be assigned by the Consumer (E.g. Tax Reference Number of an individual or Customs Code)
- **_ Underscore separator**
- **< Unique File ID >:** Unique ID assigned to the file.
- **_ Underscore separator**
- **< MessageCreateDateTime>:** Timestamp in which the file was created using CCYYMMDDTHHMMSS format.
- **_ Underscore separator**

- **< File Response Code>:** The response code to tie it back to the initial request.
- **_ Underscore separator**
- **<Node name>:** The Source application's C:D node name.

E.g. WHD_2_8880000008_67890_20111025T160508_R001_ITSolutions.txt

5.3.7 Quality of Service (QoS) Setup

There are no specific QoS requirements for the C:D protocol; however the parameters in the table below must be defaulted:

Parameter	Default Value
checksumMethod	"MD5"

Table 11: Configurable Connect Direct QoS Parameters

5.3.8 Error Handling

This will be handled by the SARS Connect:Direct Technical Support team.

5.3.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.

5.4 Sterling File Gateway

5.4.1 Overview

Sterling File Gateway (SFG) is a centralized gateway that offers a scalable architecture that allows the user to administer inbound and outbound traffic for high volume files, and allows the organization to monitor the traffic. Being a centralized system, Sterling File Gateway allows for standardization of file transfer approaches, and consolidation of disparate file transfer activities in order to streamline and simplify the file transfer process.

5.4.2 Key Characteristics:

- Improved management of file transfer activity;
- Security-rich file transfers (single and batch) with data encryption.

5.4.3 Usage and Exclusions:

The following are some considerations regarding the usage of SFG:

- SFG is the alternative protocol for external organisation to integrate with SARS.
- SFG is used when an external organisation does not have access to Connect:Direct infrastructure.

5.4.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in using the Sterling File Gateway protocol:

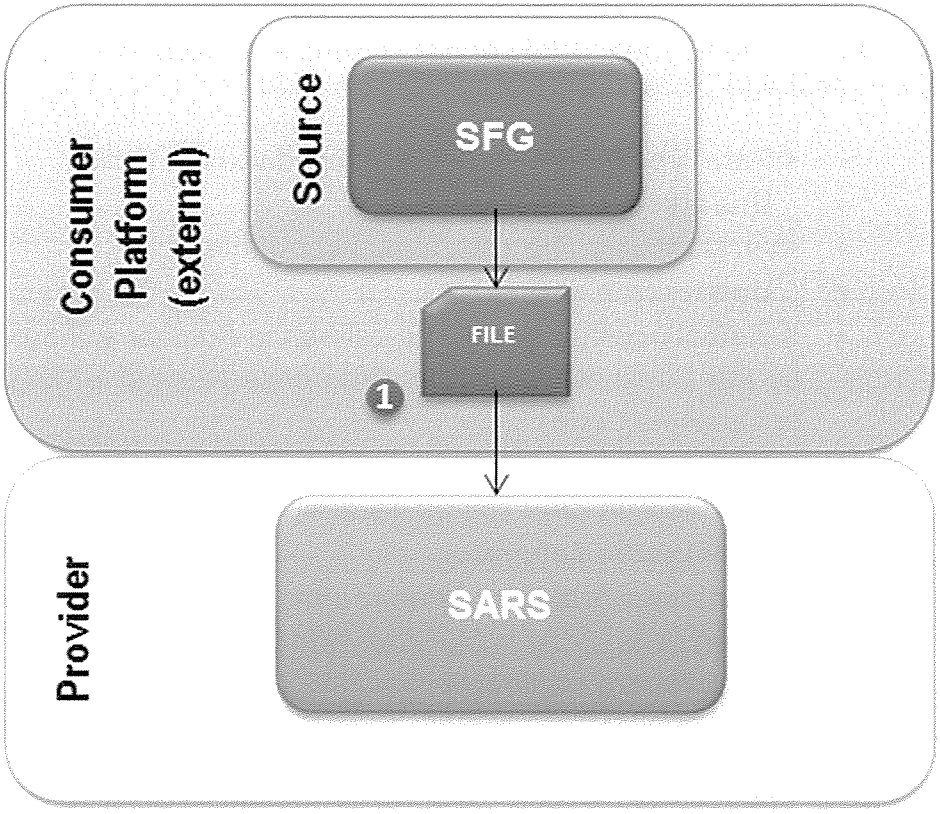


Figure 5: Typical interaction between Consumer and Provider using the Sterling File Gateway protocol

The step involved is as follows:

1. The consumer builds a request message and sends it to SARS. SARS receives the message and invokes the corresponding process.

NB: The interaction when the application acts a provider/destination follows the same steps but in reverse order (i.e. exposed via SARS).

5.4.5 Correlation

The correlation for SFG is the same as that of MQ. Please see **section 5.2.5** above.

5.4.6 Naming Convention

The file naming convention below applies to both request and response files:

5.4.6.1 The Request File

Format: <DataType>_<VersionNr>_<ReferenceNumber>_<Unique File ID>_<MessageCreateDateTime>_<Node name>

- **< DataType>**:Type of data contained within the file
- **_ Underscore separator**
- **<VersionNr >:** The version number of the file.
- **_ Underscore separator**
- **< ReferenceNumber>:** The reference number will be assigned by the Consumer (E.g. Tax Reference Number of an individual or Customs Code)
- **_ Underscore separator**
- **< Unique File ID >:** Unique ID assigned to the file.
- **_ Underscore separator**
- **< MessageCreateDateTime>:** Timestamp in which the file was created using CCYYMMDDTHHMMSS format.
- **_ Underscore separator**
- **<Node name>:** The Source application's SFG node name.

E.g. WHD_2_8880000008_67890_20111025T160508_ITSolutions.txt

5.4.6.2 The Response File

Format: <DataType>_<VersionNr>_<ReferenceNumber>_<Unique File ID>_<MessageCreateDateTime>_ < File Response Code>_<Node name>

- **< DataType>**:Type of data contained within the file e.g. WHD
- **_ Underscore separator**
- **<VersionNr >:** The version number of the file.
- **_ Underscore separator**
- **< ReferenceNumber>:** The reference number will be assigned by the Consumer (E.g. Tax Reference Number of an individual or Customs Code)
- **_ Underscore separator**
- **< Unique File ID >:** Unique ID assigned to the file.
- **_ Underscore separator**
- **< MessageCreateDateTime>:** Timestamp in which the file was created using CCYYMMDDTHHMMSS format.
- **_ Underscore separator**

- **< File Response Code>:** The response code to tie it back to the initial request.
- **_ Underscore separator**
- **<Node name>:** The Source application's SFG node name.

E.g. WHD_2_8880000008_67890_20111025T160508_R001_ITSolutions.txt

5.4.7 Quality of Service (QoS) Setup

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The SFG QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Synchronous
Delivery Assurance	N/A
Availability	24 / 7
Expiration Time	60s
Timeout	60s
Transactional (Y / N)	N
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Throughput	100 per second

Table 12: Configurable SFG QoS Parameters

5.4.8 Error Handling

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.4.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.

- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.

5.5 Secure File Transfer Protocol (SFTP)

5.5.1 Overview

The SSH File Transfer Protocol (also known as Secure FTP and SFTP) is a file transfer protocol for moving files between applications. SFTP uses encryption to encrypt contents of the file and prevent sensitive information from being transmitted in the clear over a network.

5.5.2 Key Characteristics:

- Platform and Language Independence; and
- Requires no additional technology/product investment to use.

5.5.3 Usage and Exclusions:

The following are some considerations regarding the usage of SFTP:

- SFTP is the alternative protocol for external organisation to integrate with SARS.
- Usage of SFTP should be limited to external organisations that cannot invest in any of the above methods and treated on exception basis..

5.5.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in a role of the transfer source using the SFTP protocol:

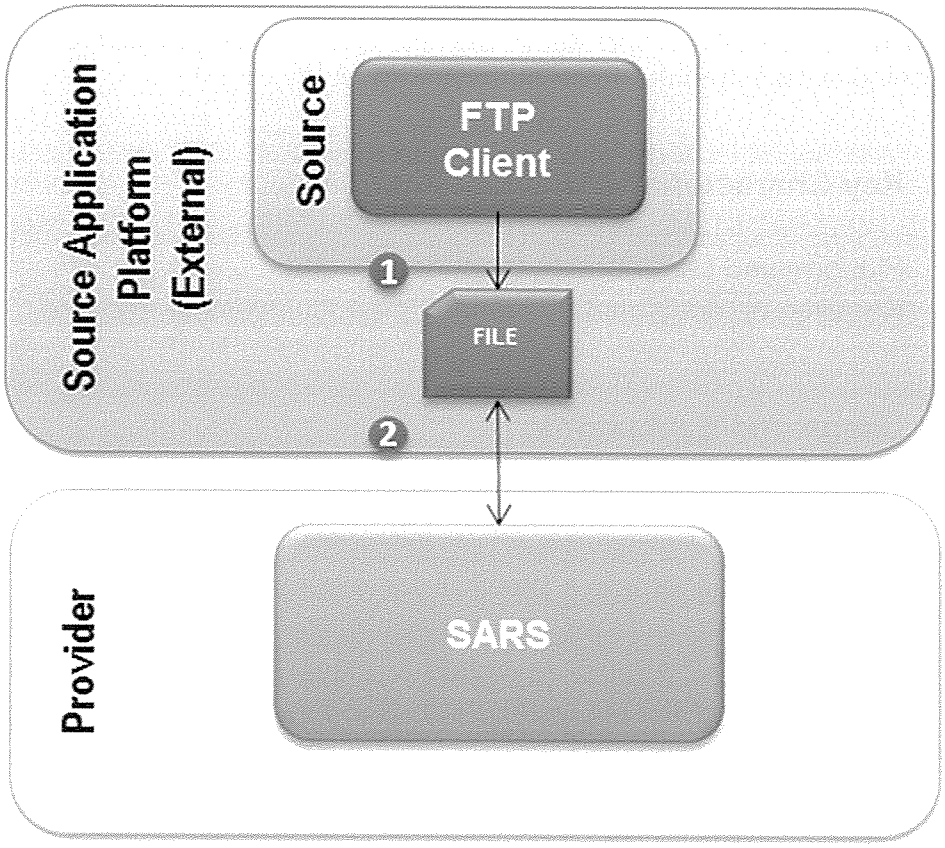


Figure 6: Typical interaction between an Application and SARS using SFTP protocol

The steps involved are as follows:

1. The external file transfer source generates the file and places it locally in a directory.
2. SARS polls the directory, picks up the file and invokes the corresponding process.

NB: The interaction when the application acts as a destination follows the same steps but in reverse order (i.e. receives a file from the DataPower device).

5.5.5 Correlation:

There is no specific correlation requirements defined for the SFTP protocol and as such, it should be handled as part of the file content.

5.5.6 Naming Convention:

The file naming convention below applies to both request and response files:

5.5.6.1 The Request File

Format: <DataType>_<VersionNr>_<ReferenceNumber>_<Unique File ID>_<MessageCreateDateTime>_<Node name>

- **< DataType>**:Type of data contained within the file e.g. WHD
- **_ Underscore separator**
- **<VersionNr >:** The version number of the file.
- **_ Underscore separator**
- **< ReferenceNumber>:** The reference number will be assigned by the Consumer (E.g. Tax Reference Number of an individual or Customs Code)
- **_ Underscore separator**
- **< Unique File ID >:** Unique ID assigned to the file.
- **_ Underscore separator**
- **< MessageCreateDateTime>:** Timestamp in which the file was created using CCYYMMDDTHHMMSS format.
- **_ Underscore separator**
- **<Node name>:** The Source application's C:D node name.

E.g. WHD_2_8880000008_67890_20111025T160508_ITSolutions.txt

5.5.6.2 The Response File

Format: <DataType>_<VersionNr>_<ReferenceNumber>_<Unique File ID>_<MessageCreateDateTime>_ < File Response Code>_<Node name>

- **< DataType>**:Type of data contained within the file e.g. WHD
- **_ Underscore separator**
- **<VersionNr >:** The version number of the file.
- **_ Underscore separator**
- **< ReferenceNumber>:** The reference number will be assigned by the Consumer (E.g. Tax Reference Number of an individual or Customs Code)
- **_ Underscore separator**
- **< Unique File ID >:** Unique ID assigned to the file.
- **_ Underscore separator**
- **< MessageCreateDateTime>:** Timestamp in which the file was created using CCYYMMDDTHHMMSS format.
- **_ Underscore separator**

- **< File Response Code>:** The response code to tie it back to the initial request.
- **_ Underscore separator**
- **<Node name>:** The Source application's C:D node name.

E.g. WHD_2_8880000008_67890_20111025T160508_R001_ITSolutions.txt

5.5.7 Quality of Service (QoS) Setup

There are no specific QoS requirements for the C:D protocol; however the parameters in the table below must be defaulted:

Parameter	Default Value
checksumMethod	"MD5"

Table 13: Configurable Connect Direct QoS Parameters

5.5.8 Error Handling

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.5.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.

5.6 WebSphere MQ File Transfer Edition (FTE)

5.6.1 Overview

WebSphere MQ File Transfer Edition (FTE) provides a managed file transfer capability that exploits the proven reliability and connectivity of WebSphere MQ to transfer files across a wide range of platforms and networks.

5.6.2 Key Characteristics:

- Guaranteed Delivery;
- Reliable handling of large-sized files with the ability to resume; and
- Platform and Language Independent and leverages existing investment in WebSphere MQ.

5.6.3 Usage and Exclusions:

The following are some considerations regarding the usage of MQ FTE:

- MQ FTE is the preferred protocol for internal and external Batch type integration and should only be deviated from for once off file transfers (i.e. Data migration/take-on file transfers).

5.6.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in a role of a transfer source using the MQ FTE protocol:

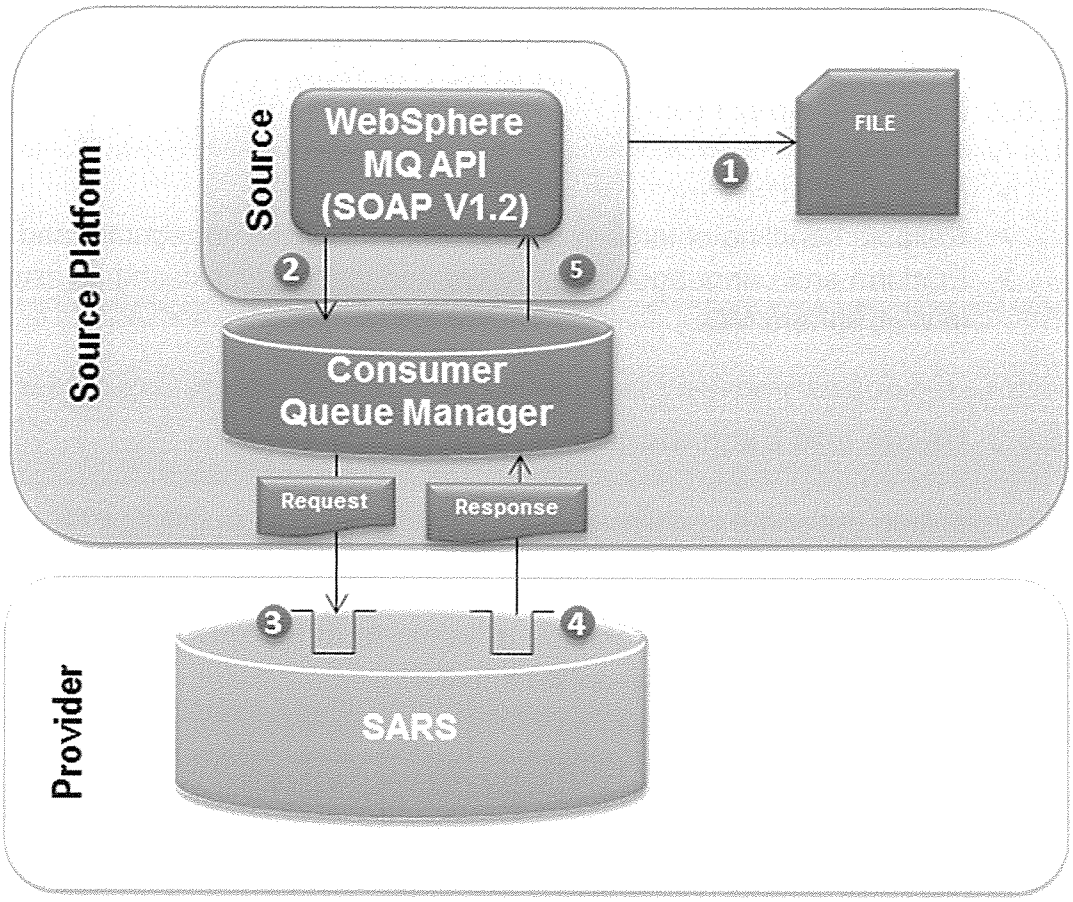


Figure 7: Typical interaction between an Application and SARS using the MQ FTE protocol:

The steps involved are as follows:

1. The File transfer source generates the file and places it locally.
2. The File transfer source builds a request message (i.e. File Transfer Request) and puts it on the designated request queue of its Queue Manager via the MQ API.
3. The MQ infrastructure moves the message to the corresponding request queue on SARS's Queue Manager.
4. SARS sends the response message to the consumer's Queue Manager.
5. The application polls the queue and reads the message via the MQ API and processes the results.

NB: The interaction when the application acts as a destination follows the same steps but in reverse order (i.e. its FTE agent receives the file from the source agent).

5.6.5 Correlation:

There is no specific correlation for the MQ FTE protocol but SARS will send a response message back to the Transfer source application (i.e. Application Information) and as such, its correlation to the initial File Transfer Request must be the same as in the MQ protocol.

5.6.6 Naming Convention:

There is no specific file naming convention except that the file name must be suffixed with either .REQ to indicate that it's a request file or .RES to indicate that it's a response file.

5.6.7 Quality of Service (QoS) Setup

There are no specific QoS requirements for the MQ FTE protocol but the following defaults apply:

Parameter	Default Value
ChecksumMethod	"MD5"
reply@AMGR (Queue Manager)	Defaulted by SARS to reply queue manager name
reply@qdef (Queue Name)	Defaulted by SARS to reply queue name
Hostname	"localhost"
User ID	"root"

Table 14: Configurable MQ FTE QoS Parameters

5.6.8 Error Handling:

There is no specific error handling requirements for MQ FTE defined yet.

5.6.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.

5.7 Java Messaging Service (JMS)

5.7.1 Overview

The Java Message Service (JMS) API is a Java Message Oriented Middleware (MOM) API for sending messages. It is a messaging standard that allows application components based on the Java Enterprise Edition (Java EE) to communicate with other components and external applications in a robust manner.

5.7.2 Key Characteristics:

The following are some of the key characteristics of JMS:

- Asynchronous, Store-and-Forward type processing;
- Guaranteed Delivery;
- Performance and Ability to handle large volumes; and
- Platform independent but specific to Java based applications.

5.7.3 Usage and Exclusions:

Below are the java specification versions that need to be adhered to:

- JMS specification is JMS 1.1.
- JavaBeans Activation Framework (JAF) Specification is JAF 1.1.

The following are some considerations regarding the usage of the JMS protocol:

- JMS is the standard protocol for all Java based applications as it is the standard API most familiar to Java developers.
- The Java based application must be deployed on the WebSphere Application Server (WAS) platform.
- The application must only communicate with WAS via JMS and not directly with the MQ API.
- Additionally, WAS is the standard application server on which Java applications must be hosted.
- WAS handles the communication with the ESB via the SI BUS.
- All SI BUS queues should be accessed via JMS JNDI (Java Naming and Directory Interface) lookups, and no SI BUS queue should ever be accessed directly from the application.
- Message driven beans together with Activation Specifications (JavaBeans Activation Framework (JAF) Specification) are the recommended implementation method to consume messages from a JMS provider.

5.7.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in a role of a transfer source using the JMS protocol:

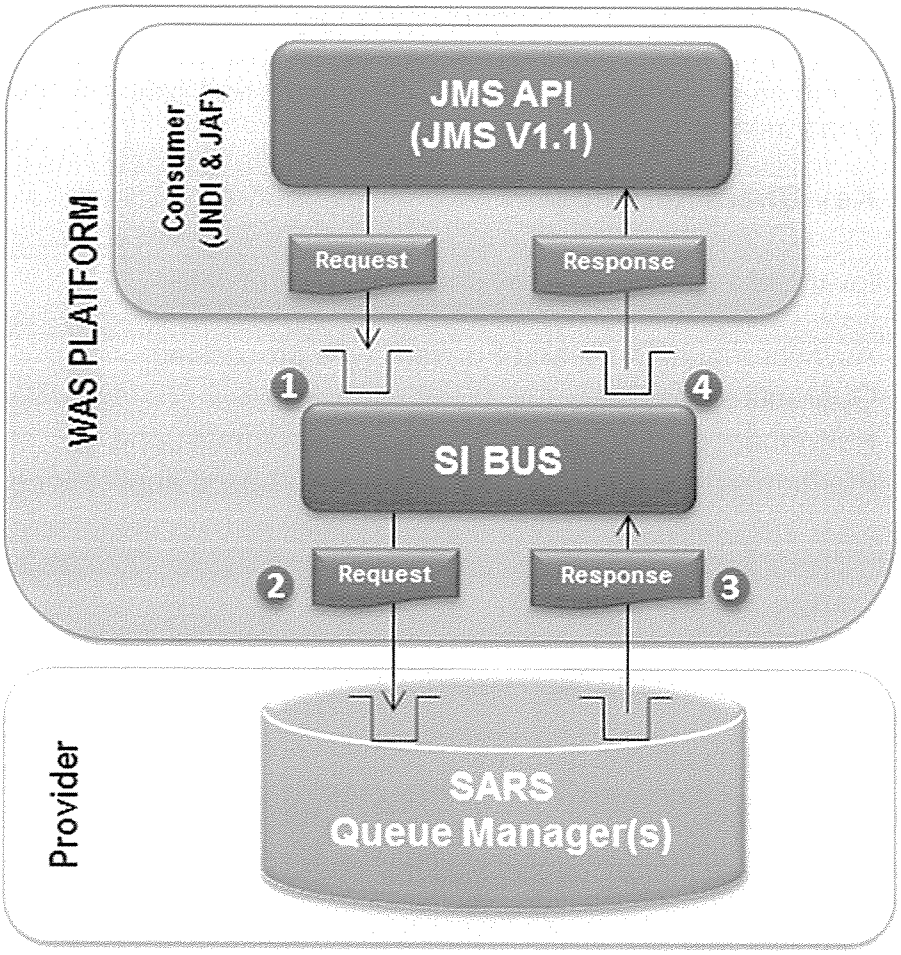


Figure 8: Typical interaction between an Application and SARS using the JMS protocol

The steps involved are as follows:

1. The consumer builds a request message and puts it on the WAS infrastructure via the JMS API.
2. The SI BUS moves the message to the corresponding request queue on the SARS's Queue Manager.
3. SARS reads the message off the queue, processes it and then sends the response message to the response queue on the WAS infrastructure.
4. The application polls the queue and reads the message via the JMS API and processes the results.

NB: The interaction when the application acts a provider/destination follows the same steps but in reverse order (i.e. read from request queue and put on response queue).

5.7.5 Correlation:

The correlation for JMS is the same as that of MQ with the exception of using the JMS message header & fields and setting the Correlation ID and not the Message ID.

5.7.6 Naming Convention:

The naming convention below applies to both request and response queues and will be supplied by SARS in the Logical Integration Design (LID) document.

Format: *<Channel Prefix>.<Service name>.<Queue type>*

- **<Channel Prefix>:** Application's Channel ID
- **. Dot separator**
- **<Service name>:** This name describes the function(s) to be performed by the service.
- **. Dot separator**
- **<Queue type>:** Specified as REQ for a request message and RES for a response message

E.g. EXT.SAMPLESERVICE.REQ
EXT.SAMPLESERVICE.RES

5.7.7 Quality of Service (QoS) Setup

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The JMS QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Asynchronous
Delivery Assurance	Guaranteed Delivery
Availability	24 / 7
Expiration Time	7 Days
Timeout	60s (Front-end only)
Transactional (Y / N)	Y
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Throughput	1000 per second

Table 15: Configurable JMS QoS Parameters

5.7.8 Error Handling:

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.7.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.

5.8 Advanced Message Queuing Protocol (AMQP)

5.8.1 Overview:

Advanced Message Queuing Protocol (AMQP) is an open source standard for passing business messages between applications and/or organizations. It creates a full functional interoperability between conforming clients and messaging middleware servers/brokers.

5.8.2 Key Characteristics:

The following are some of the key characteristics of MQ:

- Asynchronous;
- Guaranteed Delivery - from Fire and forget to reliable, exactly once acknowledged delivery;
- Any client can initiate communication with and then communicate with any AMQP broker over TCP/IP;
- Accommodates existing messaging API standards;
- Can run on multiple operating systems and CPU architectures; and
- Interoperable cross platform messaging standard.

5.8.3 Usage and Exclusions:

The following are some considerations regarding the usage of AMQP:

- Integrate with organisations that do not have IBM WebSphere MQ;
- Lower cost of enterprise and systems integration; and
- Use AMQP to deliver on Service Oriented Architecture.

5.8.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in a role of a service consumer using the AMQP protocol:

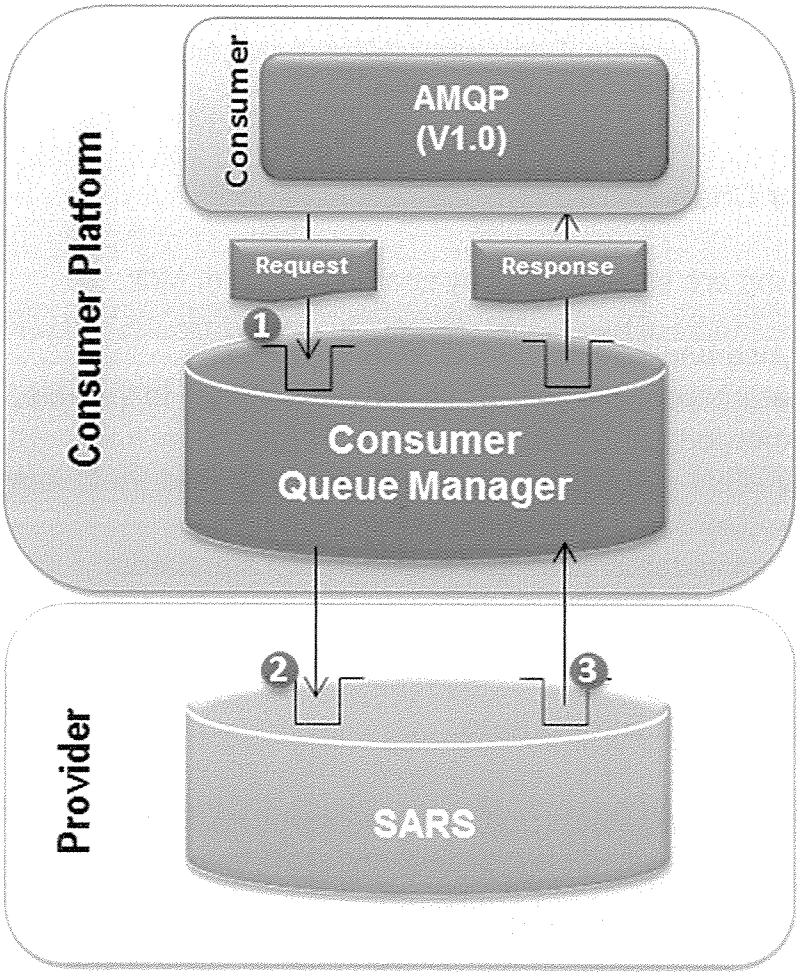


Figure 9: Typical interaction between an Application and SARS using the AMQP protocol

The steps involved are as follows:

1. The consumer builds a request message and puts it on the designated request queue of its Queue Manager via AMQP.
2. The AMQP infrastructure moves the message to the corresponding request queue on the SARS Queue Manager.
3. SARS sends the response message to the consumer's Queue Manager.

NB: The interaction when the application acts a provider/destination follows the same steps but in reverse order (i.e. read from request queue and put on response queue).

5.8.5 Correlation:

The message header will be used to correlate the response to the request message.

5.8.6 Naming Convention:

The naming convention below applies to both request and response queues and will be supplied by SARS in the Logical Integration Design (LID) document.

Format: *<Channel Prefix>.<Service name>.<Queue type>*

- **<Channel Prefix>:** Application's Channel ID
- **. Dot separator**
- **<Service name>:** This name describes the function(s) to be performed by the service.
- **. Dot separator**
- **<Queue type>:** Specified as REQ for a request message and RES for a response message

E.g. EXT.SAMPLESERVICE.REQ
EXT.SAMPLESERVICE.RES

5.8.7 Quality of Service (QoS) Setup:

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The AMQP QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Asynchronous
Delivery Assurance	Guaranteed Delivery
Availability	24 / 7
Expiration Time	7 Days
Timeout	60s (Front-end only)
Transactional (Y / N)	Y
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Preferred Size	Less than 10MB
Throughput	1000 per second

Table 16: Configurable AMQP QoS Parameters

5.8.8 Error Handling:

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.8.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.

5.9 EDI Gateway Protocols

5.9.1 AS2

5.9.1.1 Overview

Applicability Statement 2 (AS2) is a specification about how to transmit UN/EDIFACT data securely and reliably over the internet.

5.9.1.2 Key Characteristics

- AS2 is HTTP-based;
- Messages can be encrypted; and
- Synchronous and Asynchronous processing.

5.9.1.3 Usage and Exclusions:

The following are some considerations regarding the usage of the AS2 protocol:

- The usage of AS2 allows seamless transmission of EDI data so that it is readily available in real time;
- Uses digital certificates to ensure that documents are delivered only to intended recipients(SARS Public key will be sent to the Trading Partner);

5.9.1.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in using the AS2 protocol:

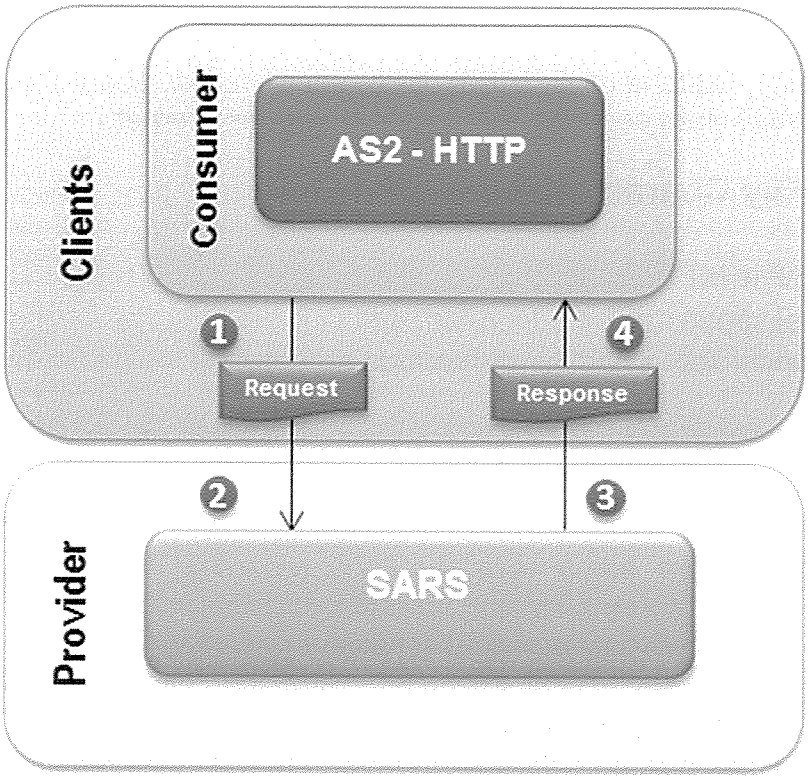


Figure 10: Typical interaction between Consumer and Provider using the AS2 protocol

The steps involved are as follows:

1. The consumer builds a request message and sends it to SARS via the AS2 Protocol over HTTP.
2. The input node receives the message and invokes the corresponding process.
3. A response is sent to the consumer.
4. This AS2 call is concluded and the response returned to the consumer.

5.9.1.5 Correlation

There is no specific correlation requirements defined for AS2 protocol.

5.9.1.6 Naming Convention

The file naming convention below applies to the incoming files:

Format: <DateTime><Counter>.<Extension>

- <DateTime>: Date timestamp when file was created using YYYYMMDDHHMM format.
- <Counter>:Number counter beginning at 0001-9999.
- <Extension>: .txt

e.g. 2017092712330001.txt

For outgoing files, the filename is made up of the unique identifier which is generated by SARS, and has a file extension .edi

5.9.1.7 Quality of Service (QoS) Setup

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The Web service QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Synchronous
Delivery Assurance	N/A
Availability	24 / 7
Expiration Time	60s
Timeout	60s
Transactional (Y / N)	N
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Throughput	100 per second

Table 17: Configurable AS2 QoS Parameters

5.9.1.8 Error Handling

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.9.1.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.
- SARS will not manipulate data.

5.9.2 AS3

5.9.2.1 Overview

Applicability Statement 3 (AS3) is a specification about how to transmit UN/EDIFACT data securely and reliably using the FTP protocol.

5.9.2.2 Key Characteristics

- AS3 is FTP-based;
- Asynchronous; and
- Messages can be encrypted.

5.9.2.3 Usage and Exclusions:

The following are some considerations regarding the usage of the AS3 protocol:

- The usage of AS3 allows seamless transmission of EDI data so that it is readily available in real time;
- Uses digital certificates to ensure that documents are delivered only to intended recipients (SARS Public key will be sent to the Trading Partner);

5.9.2.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in using the AS3 protocol:

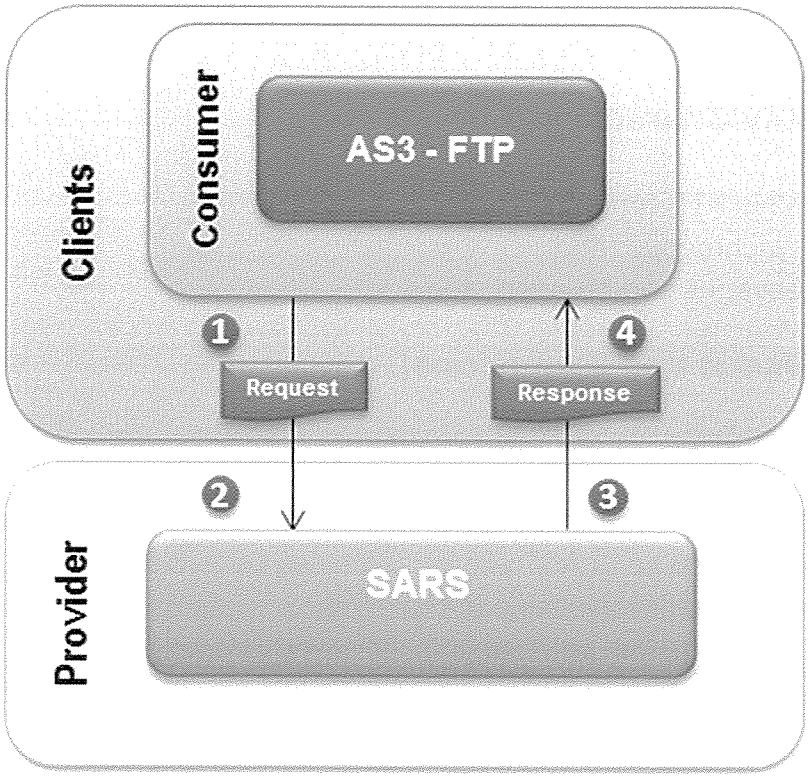


Figure 11: Typical interaction between Consumer and Provider using the AS3 protocol

The steps involved are as follows:

1. The consumer builds a request message and sends it to SARS via the AS3 FTP Protocol.
2. The input node receives the message and invokes the corresponding process.
3. A response is sent to the consumer.
4. This AS3 call is concluded and the response returned to the consumer.

5.9.2.5 Correlation

There is no specific correlation requirements defined for AS3 protocol.

5.9.2.6 Naming Convention

The file naming convention below applies to the incoming files:

Format: <DateTime><Counter>.<Extension>

- <DateTime>: Date timestamp when file was created using YYYYMMDDHHMM format.
- <Counter>:Number counter beginning at 0001-9999.
- <Extension>: .txt

The Trader places the file in the "inbox" on the AS3 server.

e.g. 2017092712330001.txt

For outgoing files, the filename is made up of the unique identifier which is generated by SARS and is placed directly in the relevant Traders directory on the AS3 server.

5.9.2.7 Quality of Service (QoS) Setup

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The Web service QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Asynchronous
Delivery Assurance	N/A
Availability	24 / 7
Expiration Time	60s
Timeout	60s
Transactional (Y / N)	N
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Throughput	100 per second

Table 18: Configurable AS3 QoS Parameters

5.9.2.8 Error Handling

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.9.2.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.
- SARS will not manipulate data.

5.9.3 X.400

5.9.3.1 Overview

X.400 is a suite of protcols defining standards for email messaging systems. Defined by the ITU-TS (International Telecommunications Union – Telecommunications Sector). Used as an alternative to the more common email protocol called Simple Mail Transfer Protocol (SMTP).

5.9.3.2 Key Characteristics

- Low-Cost solution;
- Simple;
- Generic message protocol extensibility; and
- Messaging in secure environments.

5.9.3.3 Usage and Exclusions:

The following are some considerations regarding the usage of the X.400 protocol:

- The usage of X.400 allows seamless transmission of EDI data so that it is readily available;

5.9.3.4 Typical Interaction Architecture:

The diagram below illustrates the interaction of an application communicating with SARS in using the X.400 protocol:

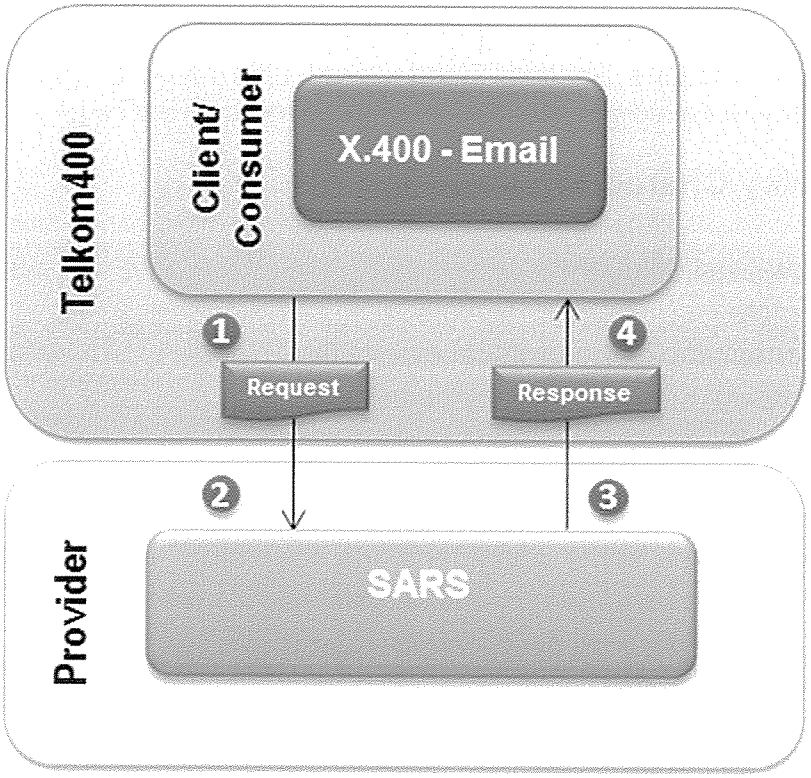


Figure 12: Typical interaction between Consumer and Provider using the X.400 protocol

The steps involved are as follows:

1. The client/consumer builds a request message and sends it to SARS MTA at Telkom400.
2. The input node receives the message and invokes the corresponding process.
3. A response is sent to Telkom400 MTA.
4. This X.400 call is concluded and the response returned to the consumer.

5.9.3.5 Correlation

There is no specific correlation requirements defined for X.400 protocol.

5.9.3.6 Naming Convention

The file naming convention below applies to the incoming files:

Format: <File_name>.<Extension>

- <File_name>: Unique identifier created by the client/consumer.
- <Extension>: .edi

5.9.3.7 Quality of Service (QoS) Setup

The design and development of each Service must involve defining and configuring communication level QoS parameters to ensure that the implemented Service meets business and operational expectations.

The Web service QoS parameters that are to be specified and their associated defaults (if not explicitly specified) are as follows:

QoS Parameter	Default Value
Synchronous/ Asynchronous/ Fire & Forget	Synchronous
Delivery Assurance	N/A
Availability	24 / 7
Expiration Time	60s
Timeout	60s
Transactional (Y / N)	N
Priority: 0 (low) – 9 (high)	5
Message Size Limit	2 MB
Throughput	100 per second

Table 19: Configurable X.400 QoS Parameters

5.9.3.8 Error Handling

Technical Error Messages are to be sent via a specified mechanism to the External Development Partner.

5.9.3.9 Message Reprocessing:

Below are the rules and responsibilities for message reprocessing:

- If a message needs to be reprocessed, the service consumer/intermediary acting as a consumer is responsible for resubmitting request messages.
- SARS will not resubmit any messages but it can facilitate the movement of messages that are already on a queue.
- SARS will not manipulate data.

6 Security

SARS has been modernising and simplifying its tax process in line with international best practices. As part of this process, SARS ensures that 3rd Party Data is protected.

For further information regarding "The Third Party Data Enrolment and Activation", click on the link below:

http://www.sars.gov.za/AllDocs/Documents/3rdPartyData/GEN-ENR-01-G01%20-%20The%20Third%20Party%20Data%20Enrolment%20-%20External%20Guide.pdf

7 Message Formats

7.1 Encoding Standard

The default encoding is UTF-8 across all messages irrespective of format and Communication Protocol.

7.2 SOAP Style XML

7.2.1 SOAP

Below is the SOAP specification version that needs to be adhered to:

- SOAP 1.2.

The Message Format Standard for SARS is the Web Services standard Simple Object Access (SOAP) Protocol message format as defined by the World Wide Web Consortium (W3C). This format has been selected due to its native support of the following:

- Independence from the Communication Protocol used;
- Formal specification for separation of header and body content; and
- Use and adherence to XML and XML Schema Definition

SOAP style messages define three basic content items that may be placed into the SOAP message: Header, Body and Fault. Although the fault element is part of the SOAP specification, it is not used except in scenarios where the protocol is Web Service and a communication-level error occurs.

7.2.2 SOAP Header

The SOAP Header is used to convey information necessary for:

- Transaction logging, tracking and reporting; and
- Message logging, routing, tracking, reporting and application-level correlation.

This is achieved through the use of a standard structure in the SOAP Header. In instances where Web Services are used, the content of the SOAP header is reserved for protocol level information.

7.2.3 SOAP Body

The SOAP Body is used to convey the business information associated with the message. The content of the Body differs depending on whether the message is a request or response message. The structure is as follows

- Request (Generated by Service Consumer):
  - Header (Web Service only)

- Response (Generated by Service Provider):
  - Header (Web Service only)
  - In instances where a functional response is required, the SOAP Body contains an optional response element. The response element is to be populated if the Provider was able to successfully process the request. The data requirement of the response element is defined in the Integration Specification of the associated service and governed and validated by the associated service schema. In the instance where a functional response is required, the Service schema will have the root the element for the response.

7.2.4 SOAP Fault

A SOAP fault is an error in a SOAP communication resulting from incorrect message format, header-processing problems, or incompatibility between applications.

When a SOAP fault occurs, a special message is generated that contains data indicating where the error originated and what caused it. This data is called a fault element. A message that includes a fault element is known as a fault message. A fault message can be generated by any node, and is transmitted to the first upstream node (the node immediately preceding it in the message path).

Below mentioned are few important points about SOAP fault element to take note of –

- A SOAP message can carry only one fault block.
- Fault is an optional part of a SOAP message.
- For HTTP binding, a successful response is linked to the 200 to 299 range of status codes.
- SOAP Fault is linked to the 500 to 599 range of status codes.

Sub-elements of Fault

The SOAP Fault has the following sub elements –

Sub-element	Description
<faultCode>	It is a text code used to indicate a class of errors. See the next Table for a listing of predefined fault codes.
<faultString>	It is a text message explaining the error.
<faultActor>	It is a text string indicating who caused the fault. It is useful if the SOAP message travels through several nodes in the SOAP message path, and the client needs to know which node caused the error. A node that does not act as the ultimate destination must include a *faultActor* element.
<detail>	It is an element used to carry application-specific error messages. The detail element can contain child elements called detail entries.