**SARS**

**USER GUIDE**

**Customs Supporting Document Solution for Trader Applications**

***VERSION: 0.1***

***REVISION DATE: 22 MARCH 2011***

CONTENTS

[1. OVERVIEW 3](#_Toc288638719)

[2. E@SYSCAN 4](#_Toc288638720)

[2.1 INTRODUCTION 4](#_Toc288638721)

[2.2 USING THE API 4](#_Toc288638722)

[2.3 USING THE COMMAND LINE 4](#_Toc288638723)

[2.4 USING AS A DESKTOP APPLICATION OR AN APPLET 5](#_Toc288638724)

[3. E@SYPACKAGER 12](#_Toc288638725)

[3.1 INTRODUCTION 12](#_Toc288638726)

[3.2 USING THE API 12](#_Toc288638727)

[3.3 USING THE COMMAND LINE 12](#_Toc288638728)

[4. CONTENT MANAGEMETN UPLOAD (WEB SERVICE) 14](#_Toc288638729)

[4.1 INTRODUCTION 14](#_Toc288638730)

[4.2 USING THE API 14](#_Toc288638731)

# OVERVIEW

To Be Completed ...

# E@SYSCAN

## INTRODUCTION

The e@syScan solution was developed to facilitate the scanning of electronic supporting documents to SARS. Using the provided library will enable the Service Providers to integrate document scanning capability, that conforms to SARS standards, as part of their solution.

## USING THE API

***\*\*Please consult the relevant JavaDoc for a reference to the API available for e@syScan.***

## USING THE COMMAND LINE

Currently 3 command line options are available for use, they are:

**Option 1 – scanAndConvert**

This option is used to scan a document from an available TWAIN source.

Java –jar easyscan.jar scanAndConvert <Output File> [TWAIN DSN] <Document Type>

|  |  |  |
| --- | --- | --- |
| **Argument** | **Description** | **Example** |
| Output File | The absolute path to the file that needs to be saved | C:\easyPackager\input\invoice.pdf |
| TWAIN DSN | The TWAIN data source name to be used for scanning. Please note that this needs to be preconfigured on the machine | If you supply an empty string the application will ask for a datasource from the user. |
| Document Type | The Document Type tag for the document scanned. List provided by SARS | Import – Invoice – refer a section where we explain the types |

**Option 2 - convert**

This option is used to convert a document to the prescribed SARS PDF format and also to apply document compression, where possible.

java –jar easyscan.jar convert <Input file> <Output File> <Document Type>

|  |  |  |
| --- | --- | --- |
| **Argument** | **Description** | **Example** |
| Input File | The absolute path to the file that needs to be converted | C:\easyPackager\input\input.xls |
| Output File | The absolute path to the file that needs to be saved | C:\easyPackager\output\output.pdf |
| Document Type | The Document Type tag for the document scanned. List provided by SARS | Import - Invoice refer a section where we explain the types |

**Option 3 - customScanAndConvert**

This option is used to scan a document from an available TWAIN source. It is the same as Option 1 with the added option to specify the DPI setting for scanning.

java –jar easyscan.jar customScanAndConvert <Output File> <DPI> [TWAIN DSN] <Document Type>

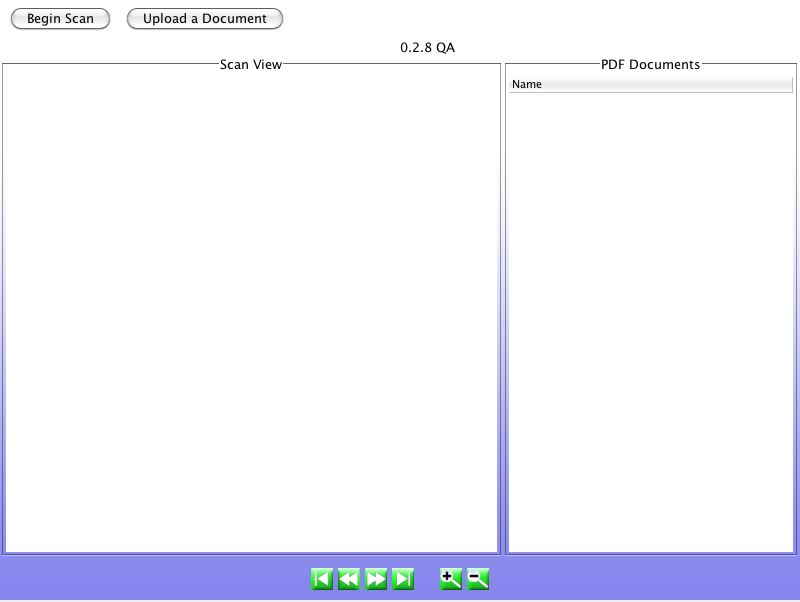
|  |  |  |
| --- | --- | --- |
| **Argument** | **Description** | **Example** |
| Output File | The absolute path to the file that needs to be saved | C:\easyPackager\input\invoice.pdf |
| DPI | DPI setting for scanning | 300 |
| TWAIN DSN | The TWAIN data source name to be used for scanning. Please note that this needs to be preconfigured on the machine | If you supply an empty string the application will ask for a datasource from the user. |
| Document Type | The Document Type tag for the document scanned. List provided by SARS | Import - Invoice refer a section where we explain the types |

## USING AS A DESKTOP APPLICATION OR AN APPLET

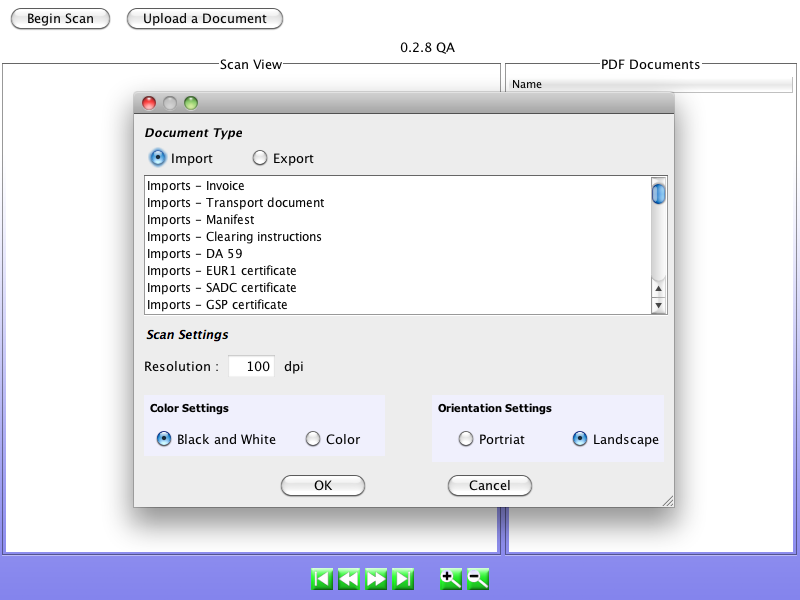
## Screen shot 2011-03-22 at 3.10.56 PM.png

* + 1. **Scanning**

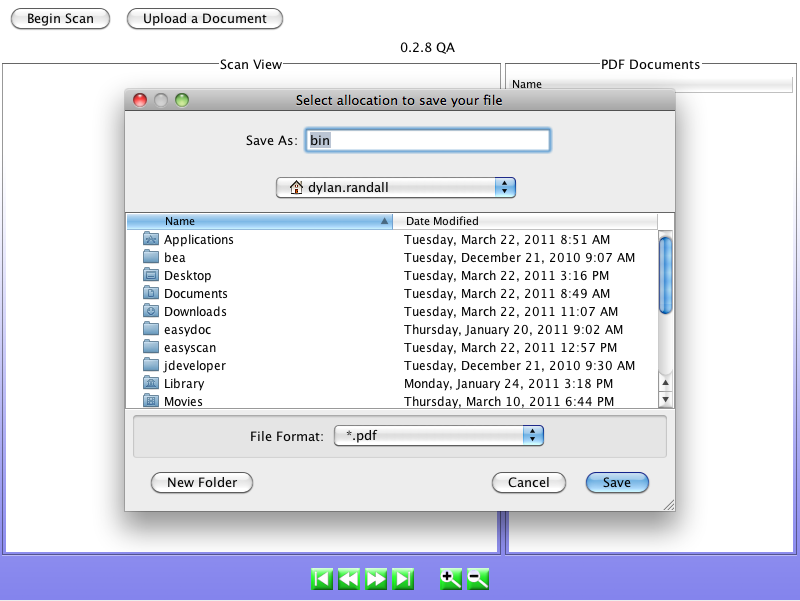
Step 1: Click Scan



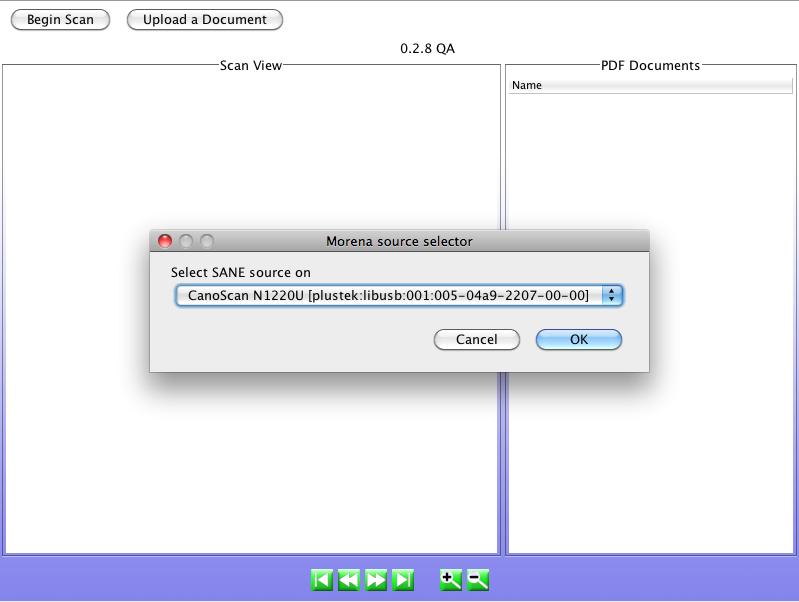
STEP 2: Begin Scan



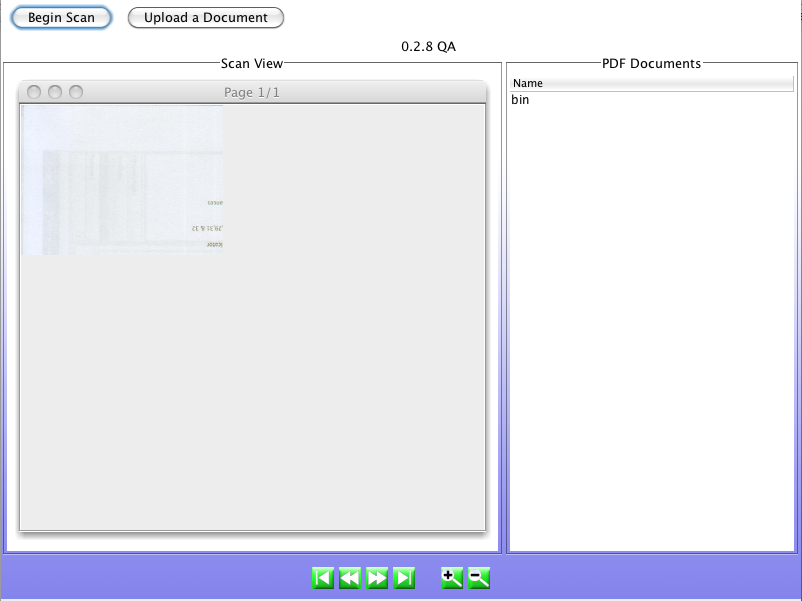
Step 3: Select where to save file



Step 4: select souce to scan

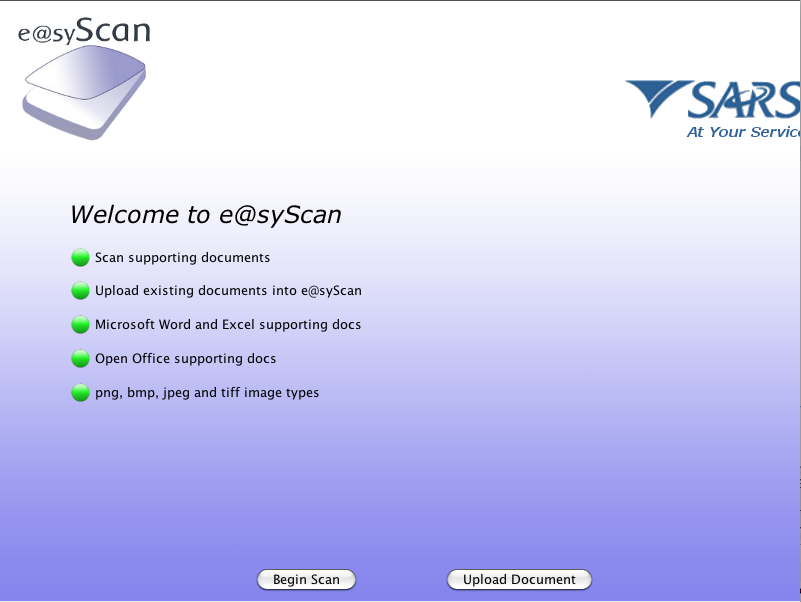


Step 5: view

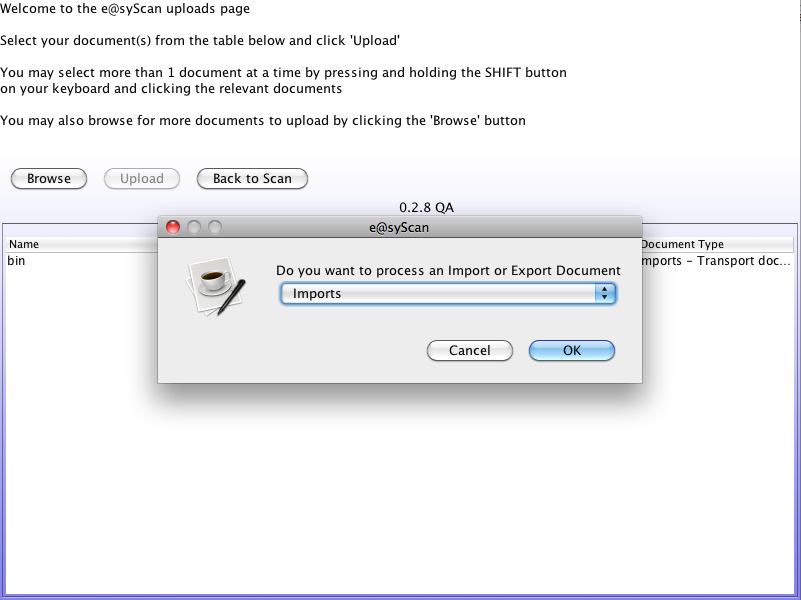


* + 1. **Upload**

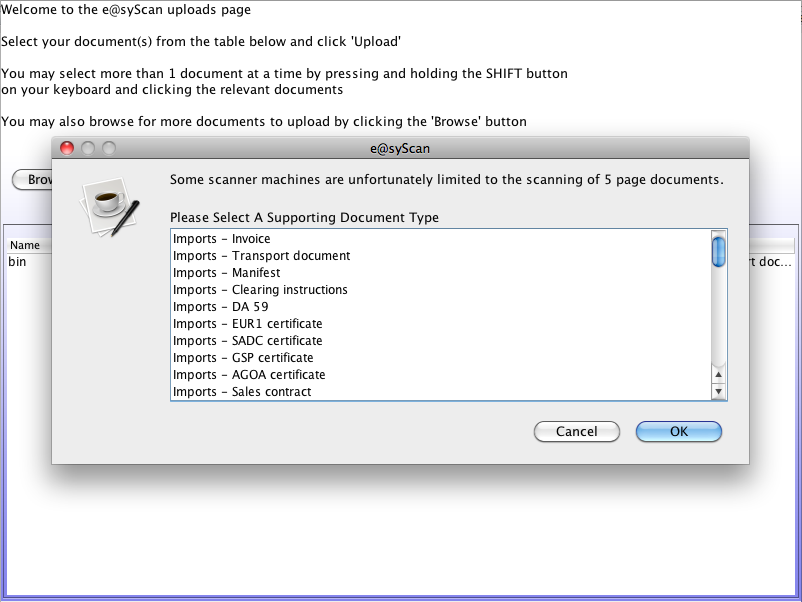
Step 1: click ‘Upload Document’



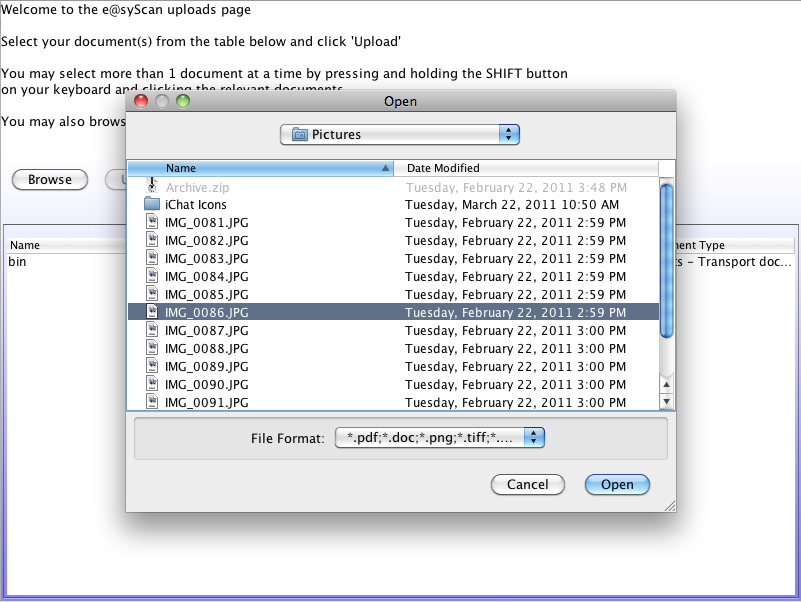
Step 2: click ‘Browse’



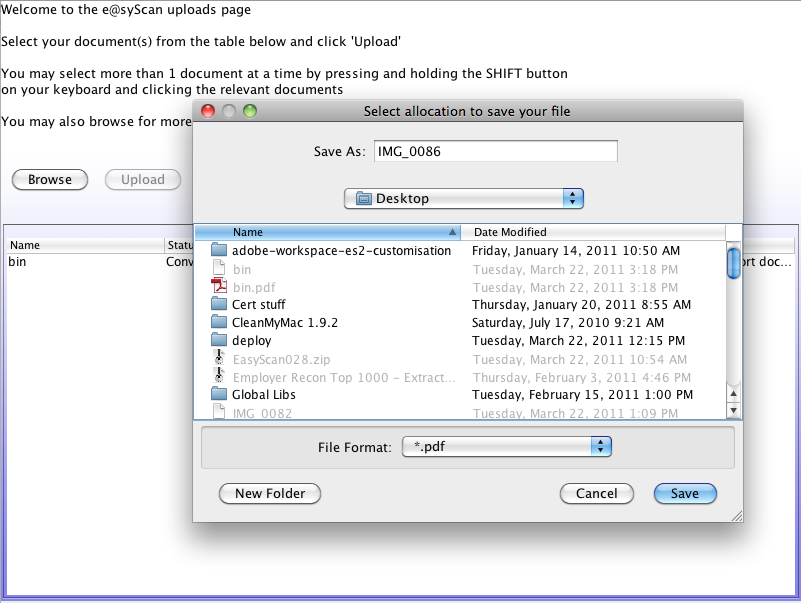
Step 3: select doc type



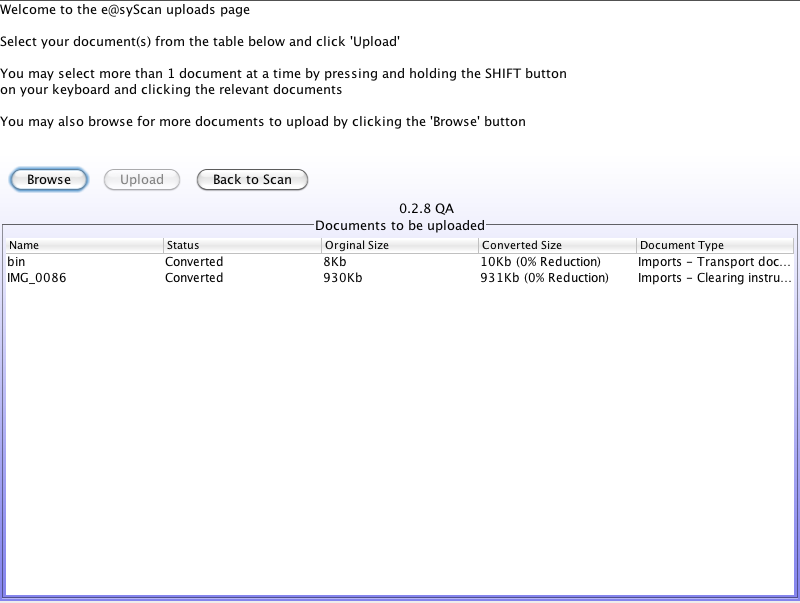
Step 4: select file to convert



Step 5: Select where to save file



Step 6: view converted document



# E@SYPACKAGER

## INTRODUCTION

The e@syPackager solution was developed to facilitate the upload of electronic supporting documents to SARS via any of the available Trader applications. Using the provided library will enable the Service Providers to integrate document packaging capability as part of their solution.

Please note that e@syPackager does not upload the documents, it converts the electronic documents into a BASE64 encoded XML package that can be uploaded via the Supporting Document web service.

## USING THE API

***\*\*Please consult the relevant JavaDoc for a reference to the API available for e@syPackager.***

## USING THE COMMAND LINE

Currently 3 command line options are available for use, they are:

**Option 1 – readDataFile**

Java –jar easypackager.jar readDataFile [Input File] [Output Directory] [Log Directory]

|  |  |  |
| --- | --- | --- |
| **Argument** | **Description** | **Example** |
| Input File | The absolute path to the file that needs to be converted and packaged for upload to SARS. | C:\easyPackager\input\invoice.properties |
| Output Directory | The absolute path to the output directory where the output file will be written. Please note that e@syPackager will generate a filename. | C:\easyPackager\output\ |
| Log Directory | The absolute path to the directory where the log file will be written. | C:\easyPackager\logs\ |

**Option 2 - readXMLDataFile** ***\*\*Recommended\*\****

Java –jar easypackager.jar readXMLDataFile [Input File] [Output Directory] [Log Directory]

|  |  |  |
| --- | --- | --- |
| **Argument** | **Description** | **Example** |
| Input File | The absolute path to the file that needs to be converted and packaged for upload to SARS. | C:\easyPackager\input\invoice.xml |
| Output Directory | The absolute path to the output directory where the output file will be written. Please note that e@syPackager will generate a filename. | C:\easyPackager\output\ |
| Log Directory | The absolute path to the directory where the log file will be written. | C:\easyPackager\logs\ |

**Option 3 - readXMLDataString *\*\*Recommended\*\****

Java –jar easypackager.jar readXMLDataString [XML String] [Output Directory] [Log Directory]

|  |  |  |
| --- | --- | --- |
| **Argument** | **Description** | **Example** |
| XML String | The XML String to converted and packaged |  |
| Output Directory | The absolute path to the output directory where the output file will be written. Please note that e@syPackager will generate a filename. | C:\easyPackager\output\ |
| Log Directory | The absolute path to the directory where the log file will be written. | C:\easyPackager\logs\ |

# CONTENT MANAGEMETN UPLOAD (WEB SERVICE)

## INTRODUCTION

The Docuemnt Management Web Service is the final step in submitting electronic supporitn documents to SARS. Please note that this web service needs to be used in conjunction with e@syScan and e@syPackager.

*Only a valid e@syPackager package will be accepted for upload.*

## USING THE API

To upload supporting documents to SARS you are required to do the following:

You need to connect via a web service interface and this is declared at the following url (destination):

<https://secure.qa.offline.sacustoms.co.za/CustomsMessageBroker/services/SupportingDocumentV10?wsdl>

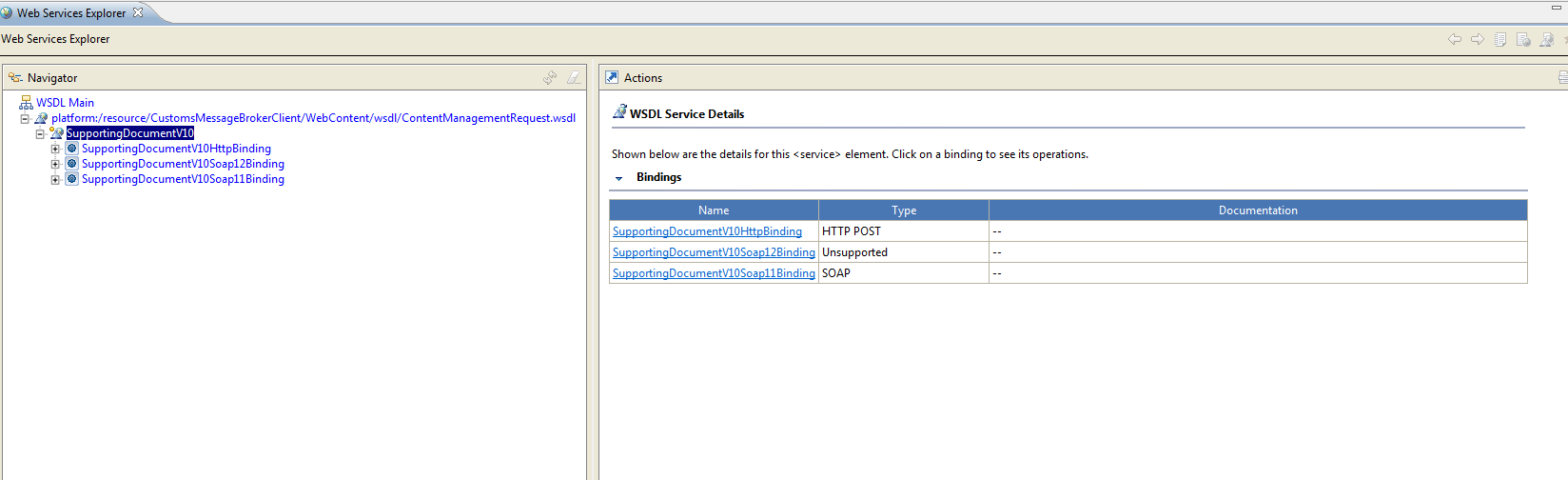
The web service exposes 3 end-point\protocols to submit data. They are (see below):

SOAP 1.1 (Supported)

SOAP 1.2 (Will be supported in future releases)

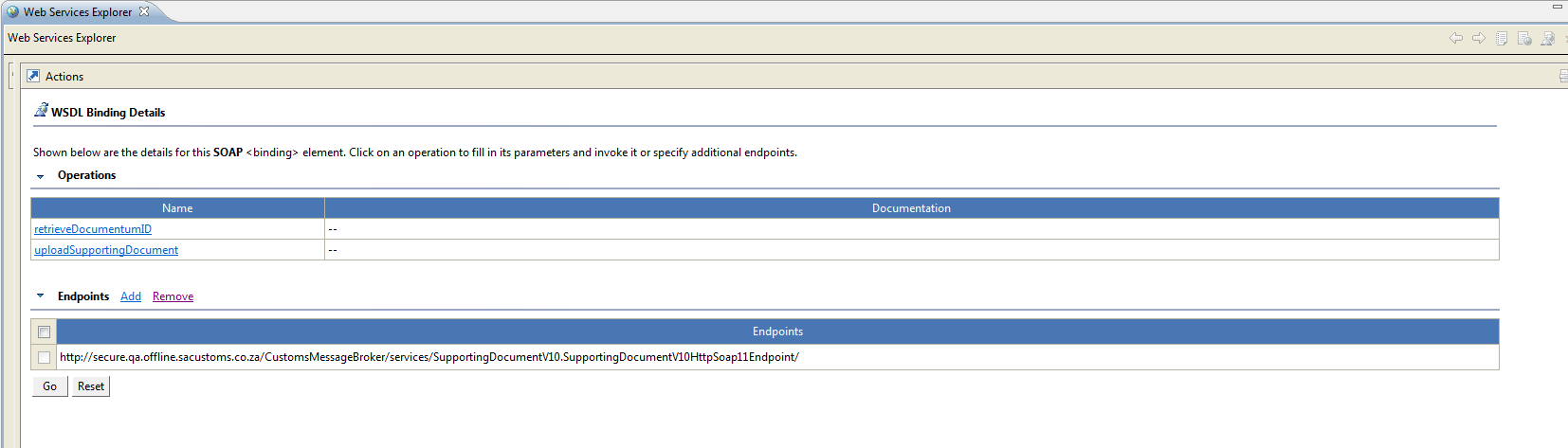
Http (Supported)

Figure 1.1 – Below is a screenshot showing the different protocols. Currently Soap 1.1 and Http is supported.



The web service also contains 2 web methods see Figure 1.2

Figure 1.2



**uploadSupportingDocument** must be used to upload your supporting document. The method contains 5 pieces of content. They are:

1. content, which is a mandatory complex type of type ContentManagementResponse. This element is provided by e@syPackager via an api call. ContentManagementResponse has properties which the e@syFile Back-End use to enable upload into the SARS content repositories. Some of these properties include:

caseNumbers

contentType

binaryData

dateOfReceipt

sourceChannel

products

objectName

path

originatingLocation

geoLocation

1. documentNo, which is a mandatory field of type integer and contains the sequence number of the document being uploaded
2. documentTotal, which is a mandatory field of type integer and contains the total number of documents that will be uploaded
3. token, which is of type string and is used by the e@syFile back-end to link documents
4. tradeAuthentication which is a mandatory complex type of type TradeAuthenticationBean. This element needs to be attached to the web service call as it contains the authentication properties to communicate to the e@syFile Back-End. The elements are:
5. AACID: which stands for Authentication Code, Authentication Qualifier and Communication Medium Identifier. This property is a combination of these three fields which is 20 character string
6. customsCode
7. name

uploadSupportingDocument responds with a complex type StatusBean. StatusBean has the following fields:

1. statusCode which is of type string and can contain the following values:

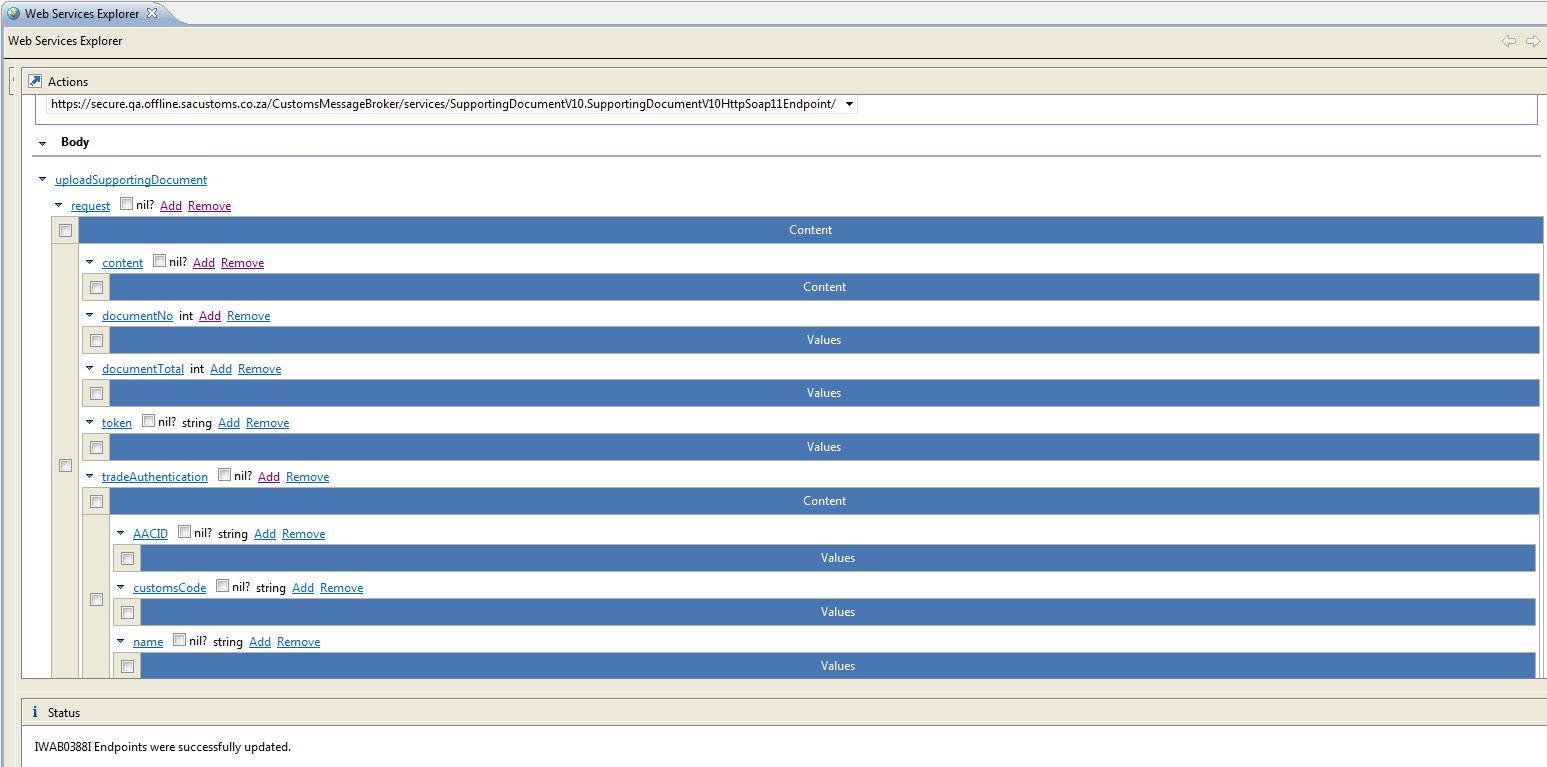
Successful

LoginFailure

Error

1. statusDescription
2. token, which is a string similar to a guid. The token will only be provided if the document has been successfully accepted and processed by the e@syFile back-end. The ‘token’ is also used for retrieval of the content repository id.

Figure 1.3 – Below shows the web method with the different types:

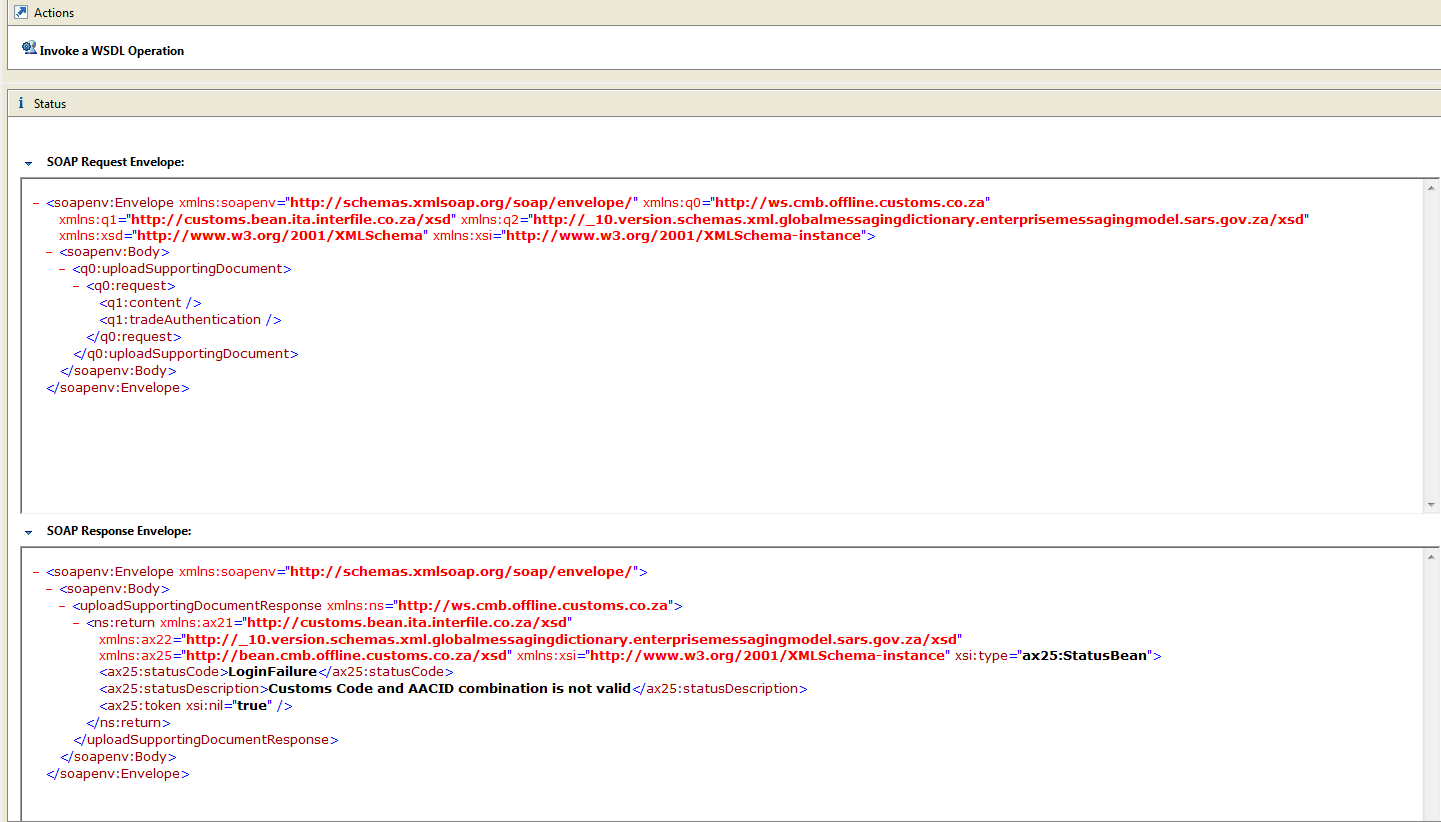


retrieveDocumentumID is the second web method and is used to retrieve the content repository id when available. retrieveDocumentumID requests that the token received on the StatusBean be passed.

Example of how a token can look like:

00000000-260e98ce-0278-4fcf-9868-16a2c1d0c409.

Example of a login failure:



Example of a request:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:q0="http://ws.cmb.offline.customs.co.za" xmlns:q1="http://customs.bean.ita.interfile.co.za/xsd" xmlns:q2="http://\_10.version.schemas.xml.globalmessagingdictionary.enterprisemessagingmodel.sars.gov.za/xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soapenv:Body>

<q0:uploadSupportingDocument>

<q0:request>

<q1:content>

<q2:binaryData>JVBERi0xLjYNJeLjz9MNCjcgMCBvYmoNPDwvTGluZWFyaXplZCAxL0wgMTY4NTI3L08gOS9FIDE2=</q2:binaryData>

<q2:dateOfReceipt>2011-02-24T15:27:35</q2:dateOfReceipt>

<q2:documentIdentifier>

<q2:type>MRNNumber</q2:type>

<q2:value>12345</q2:value>

</q2:documentIdentifier>

<q2:formID>CD1</q2:formID>

<q2:geoLocation>JHB</q2:geoLocation>

<q2:objectName>SupDoc\_DEC\_EDI\_1231232323232\_c0352530-b4a7-4ec9-8f2c-ddfe6e3d41ab</q2:objectName>

<q2:originatingLocation>JHB</q2:originatingLocation>

<q2:path>/SARS Customs Documents/EDI/2011/02/22</q2:path>

<q2:procedureCategoryCode>A</q2:procedureCategoryCode>

</q1:content>

<q1:documentNo>1</q1:documentNo>

<q1:documentTotal>1</q1:documentTotal>

<q1:tradeAuthentication>

<q1:AACID>Removed</q1:AACID>

<q1:customsCode>Removed</q1:customsCode>

<q1:name>Random Name</q1:name>

</q1:tradeAuthentication>

</q0:request>

</q0:uploadSupportingDocument>

</soapenv:Body>

</soapenv:Envelope>

Example of Response:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<soapenv:Body>

<uploadSupportingDocumentResponse xmlns:ns="http://ws.cmb.offline.customs.co.za">

<ns:return xmlns:ax21="http://customs.bean.ita.interfile.co.za/xsd" xmlns:ax22="http://\_10.version.schemas.xml.globalmessagingdictionary.enterprisemessagingmodel.sars.gov.za/xsd" xmlns:ax25="http://bean.cmb.offline.customs.co.za/xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ax25:StatusBean">

<ax25:statusCode>SUCCESSFUL</ax25:statusCode>

<ax25:statusDescription xsi:nil="true"/>

<ax25:token>00000000-a9374016-a302-4535-9a21-4854ebb0f2c2</ax25:token>

</ns:return>

</uploadSupportingDocumentResponse>

</soapenv:Body>

</soapenv:Envelope>