PLDA B2B Communication

Introduction

The purpose of this document is to describe different options that exist to electronically exchange information ("messages") with the PLDA system.

Messages can be in EDIFACT or XML format. For a number of specific exchanges the CARGOIMP format can be used, as well as a format that was developed by D&A internally (the DA format).

During the partner meetings, three options to exchange messages were adopted. These can be summarized as follows:

- SMTP message exchange
- WebService in polling mode
- WebService with callbacks

Certain reasons (among which security,reliability, consistency and the need to send back answers after a potentially longer period of time) required a new approach:

- merge both WebServices
- remove the SMTP message exchange since it was nor reliable nor secure
- require client-certificate to authenticate to link the user to a callback-address

The WebService is a service providing users with the ability, using a machine-to-machine concept, to submit Customs declaration via the Internet. This web service is provided using the SOAP protocol.

The WebService allows partners to send messages. If the partner provides a callback address, this address is used by PLDA to submit status changes and responses to. If the partner doesn't provide a callback address, the getStatus method must be polled in order to retrieve the message status lateron.

In order to properly link a message with a subsequent call for the status, the concept of a "CorrelationID" is used. This CorrelationID is a unique identifier given by the PLDA system at the first request. A CorrelationID is guaranteed to be unique for every PLDA installations (PROD, ACC, ...).

Communication

1. Web Service Description

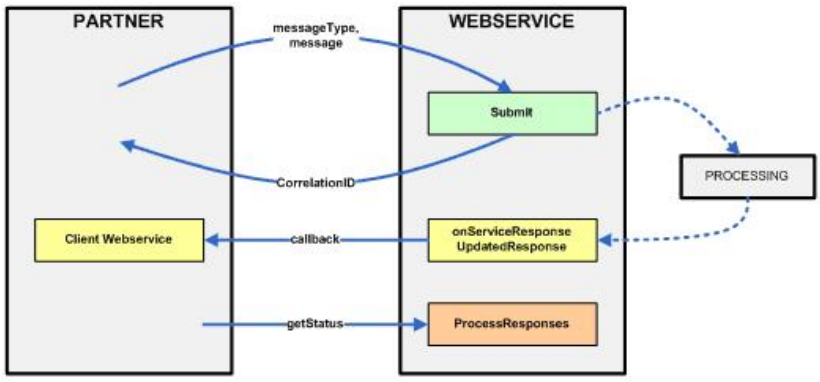
The PLDA Web Service is used to submit new Messages to Customs. You can find the wsdl here.

Multiple clients can invoke the web service simultaneously by calling the **submit** method.

The **submit** call returns a plda CorrelationID. Via the callback (this is a web service located at the client side), the client can be automatically notified of the process result. If everything is ok, the CorrelationId is not needed. But, if for a reason there is a problem with the callback, the partner can get the status of the request by using the **getStatus** and give the CorrelationID to identify the original request.

To work with callbacks, the client must register his client certificate with a callbackUrl. This callbackUrl will invoke a webservice implementing the following WSDL. The method **onServiceResponseUpdated** will be invoked on the external service with its result. It will only be called if the client has specified a callback location. If a callback is specified and the client's webservice is temporarily out of order, then the plda system will try (and retry) to send the message.

Since not all clients are capable of accepting callbacks. Specifically, clients operating from behind firewalls may not be able to receive asynchronous callbacks. For them, it is possible to not specify a callback address. This is where the polling mechanism (via **getStatus**) comes in.



Since client-certificates are required to use the webservice, it is mandatory that the client sends us

1. the client certificate that the client will use to authenticate
2. the callback on which the client wants its answers
3. an email address in case of problems

Furthermore in order to avoid going back and forth, please verify your certificate with openssl via the command:

```
openssl verify -CAfile <CAfile> -purpose sslclient <clientcert>
```

where: <CAfile> : CA-chain of your certificiate in PEM format and <clientcert>: Actual client certificate. if this return OK, you shouldn't have any problem connecting to PLDA, provided the certificate is from one of the following ca's.

- Certipost
- GlobalSign
- eTrust
- isabel

Methods Description

1. **submit**

This method makes a request to the Message Process service, by passing the Message to process. The CorrelationID is returned. If there was a problem with the request (for example, the database to store the incoming messages is unavailable) a CorrelationID of -1 is returned.

Input Parameters	
messageType	Type of message. Accepted values are: CARGOIMP, EDIFACT, XML, DA
message	message to be processed
Returns	
CorrelationID	

2. **getStatus**

This method is used to poll the Web Service to determine the status of the message. In other words, call this method to get the current process status. This method returns the ProcessResponse data array.

It is not advised to call this method too often. Typically a call every 10 seconds would be the maximum. Increasing the frequency will only put additional load on the server, while a typical message will take a number of seconds to be processed. Requesting the status more often than the typical time needed to process a message makes therefore little sense.

Input Parameters	
CorrelationID	String obtained when a call was made to the submit method
Returns	
processResponse[]	Contains process status, error,...

3. **onServiceResponseUpdated**

The method is a Callback to invoke on the client when the external service returns its result. Will only be called it the client can accept callbacks and told us where to send them. Remember that not all clients are capable of accepting callbacks.

Specifically, clients operating from behind firewalls may not be able to receive asynchronous callbacks. They should use the synchronous interface getStatus.

Input Parameters	
processResponse[]	Contains process status, error,...
Returns	
Optional string	

Type Description

The callback method onServiceResponseUpdated and the getStatus Method return a structure named **ProcessResponse**.Since more than one message could be sent in an EDIFACT transmission, the structure returned is an ARRAY.

The ProcessResponse is composed as follows:

Element	Sub Element	Description
ProcessStatus	Code	A string containing a unique status code
	Description	A text describing the status code
ProcessError	Code	A string containing a unique error code
	Description	A text describing the error code
ProcessMessage	Type	Type of Message (EDIFACT, CARGOIMP, XML)
	Name	Name of the Message (CUSCAR,...)
	Version	Version of the Message
	Release	Release of the Message
	Code	Associated code used with the message
MessageReturned	Body	Specific message returned CUSRES for EDIFACT, SADReponse for XML,...
	Type	Type of the response message (ex: cusres04A, cusdec04A,...).
CorrelationId		CorrelationId of the request message

REVISION HISTORY

Rev	Datum	Auteur	
1.0.0	25/01/05	V. Walle	Initiële Versie
1.0.1	26/01/05	R. de Schipper	Full Review
1.0.2	27/01/05	R. de Schipper	Minor changes
1.0.3	28/01/05	V. Walle	Mail Layout
1.0.4	28/01/05	R. de Schipper	Interne release
1.0.5	15/04/05	R. de Schipper	Document hernoemd. Aanpassingen mbt deployment
1.0.6	25/04/05	V. Walle	New Email Address
1.0.7	25/06/05	R. Beeckman	Minor changes
1.0.8	10/08/05	V. Walle	Webservice fusion
1.0.9	26/09/05	R. Butaye	Major changes
1.1.0	29/06/06	R. Butaye	Remove of the point 8. Security. The security is not based on username/pwd.Add example of type for webservice methods
1.1.1	11/07/06	P. Vochten	changed email address
1.1.2	21/11/06	D. Jeanfils	Change D&A by DA
1.1.3	13/12/06	R. Butaye	Change the url link to obtain the wsdl.
2.0.0	09/12/07	K. Serry	Major changes. • Require 2-way SSL to ensure security • Remove email address from parameters of submit, since we already have that information • Remove callback address as Soap Header • Rename methods to submit,getStatus • Merge both webservice (polling and callback) • Clean up documentation and convert into Html • Simplification of webservice to avoid confusion
2.0.1	09/12/07	K. Serry	Minor change: Added paragraph on how to ensure a client-certificate is valid.