Open e-PRIOR eInvoicing Integration with PEPPOL

Technical specifications

DIGIT

Directorate-General for Informatics

ISA[2] Programme

ec.europa.eu/isa2

DISCLAIMER

This document is for informational purposes only and the Commission cannot be held responsible for any use which may be made of the information contained therein. References to legal acts or documentation of the European Union (EU) cannot be perceived as amending legislation in force or other EU documentation.

The document contains a brief overview of technical nature and is not supplementing or amending terms and conditions of any procurement procedure; therefore, no compensation claim can be based of the contents of the present document.

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Commission. The European Commission does not guarantee the accuracy of the data included in this document. Neither the European Commission nor any person acting on the European Commission's behalf may be held responsible for the use which may be made of the information contained therein

Directorate-General for Informatics

ISA^2 Programme

LEGAL NOTICE

Directorate-General for Informatics

ISA^2 Programme

DOCUMENT METADATA

Property	Value
Release date:	*June 2018*
Status:	*Published in joinup*
Authors:	*George Vaicar (DIGIT B4), Grzegorz Gaudyn (DIGIT B4)*
Keywords (to increase the SEO presence)	*eProcurement, eInvoicing, PEPPOL, interoperability, e-PRIOR*

The ISA² programme supports the development of digital solutions that enable public administrations, businesses and citizens in Europe to benefit from interoperable cross-border and cross-sector public services.

ISA² is running from **1 January 2016** until **31 December 2020**. The programme was adopted in November 2015 by the European Parliament and the Council of European Union.

Solutions developed by ISA² **and under its predecessor ISA** are generally available for **free** and can be found here.

Under the ISA² programme, the following **actions are supported:** ISA² action overview page.

EXECUTIVE SUMMARY

Short Abstract (150 words)	Underlying document describes the technical aspects of the joinup release of Open e-PRIOR v2.2.0 that includes a component for integrating with the PEPPOL network, and back office eInvoicing Viewer. It can be used by any public administration to build an eInvoicing solution interoperable with the PEPPOL network.
Objectives (150 words)	Provide technical guidance on the architecture, installation and usage scenarios of the platform.
Method (300 words)	This document offers an architectural overview, a step by step installation guide as well as information on how to configure the platform and how to test it. It also includes an extension point, presenting the two options available for connecting a customer eInvoicing back-office to the platform.
Conclusions (150 words)	Technical document about the Open e-PRIOR package developed under the ISA2 programme.

AUDIENCE

This document describes the technical aspects of the joinup release of Open e-PRIOR that includes a component for integrating with the PEPPOL network and the back office eInvoicing Viewer.

It provides an architectural overview, a step by step installation guide as well as information on how to configure the platform and how to test it.

It also includes an extension point, presenting the two options available for connecting a customer eInvoicing back-office to the platform.

This document is intended for the following audiences:

AUDIENCE		TARGETED IN THIS DOCUMENT
POLICY OFFICERS		✔
IT PROVIDERS		✔
SERVICE PROVIDERS		✔
IT ARCHITECTS		✔
COMM EXPERTS		✘
LEGAL OFFICERS		✘

TABLE OF CONTENTS

Directorate-General for Informatics

ISA^2 Programme

1 Introduction and business context

1 Introduction and business context

The Directive 2014/55/EU has enforced that any public administration in Europe shall accept electronic invoices respecting the European Norm by November 2019.

In order to help administrations to meet this goal and on basis of requests of some countries, Open e-Prior has built a release that incorporates a gateway to the PEPPOL network as a first step to alignment to the e-Invoicing European Norm.

This release would provide:

- either a transitional solution for administrations not yet prepared for the Directive

- and /or a back-up solution to access received invoices

- or a solution to test the reception of PEPPOL invoices.

This release does not include anymore a supplier portal as it relies directly on the gateway and on eInvoicing Viewer.

Directorate-General for Informatics

ISA^2 Programme

2018
        p. 6 / 45
        EUR [number] EN

2 Presentation of the PEPPOL eDelivery network

The following description comes from the PEPPOL website (http://peppol.eu):

PEPPOL uses the eDelivery Network to connect different eProcurement systems by establishing a set of common business processes and technical standards. This provides an interoperable and secure network connecting all Access Points using the same electronic messaging protocol and formats and applying digital signature technologies to secure message content.

Once connected to the PEPPOL eDelivery Network (via a PEPPOL Access Point), public agencies and private enterprises can quickly and easily reach any other trading partner, also using PEPPOL.

The PEPPOL eDelivery Network's architecture is depicted in the diagram below (from https://peppol.eu/what-is-peppol/peppol-transport-infrastructure/):
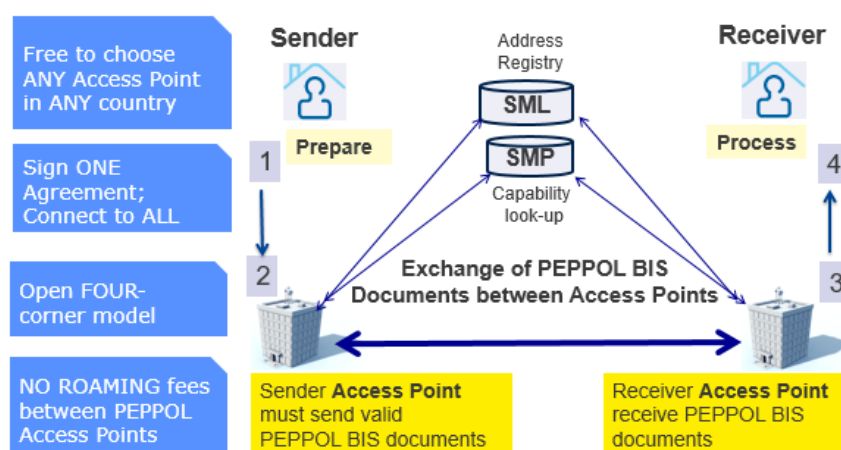
Fig. 1.   PEPPOL eDelivery Network

Source: https://peppol.eu/what-is-peppol/peppol-transport-infrastructure

The following Prezi presentation shows in more detail the components of the PEPPOL network, their role and how they interact. It also shows how the PEPPOL network interacts with the two components which are part of the current release: the PEPPOL-eTrustEx Adapter and e-TrustEx.

The presentation can be accessed here: https://prezi.com/d92orvsm2bzy/view

2 Presentation of the PEPPOL eDelivery network

2.1 Obtaining a PEPPOL certificate for the AccessPoint

The PEPPOL network uses a PKI to secure the communication between its network components.

Detailed information about the network specifications can be found on the PEPPOL website (https://peppol.eu/downloads/the-peppol-edelivery-network-specifications/)

The steps to be followed to obtain a PEPPOL certificate are also described on the PEPPOL website: https://peppol.eu/downloads/ap-guidelines/, in the step "How to enrol for OpenPEPPOL PKI certificates".

The guide includes info about how to import the new certificate into the keystore that will be referenced later in the chapters *4.2 Running the application with docker-compose* and *6.2.2 Configuring the standalone client*

3

High level architecture view

3 High level architecture view

The objective of this system is to allow the reception of UBL Invoices and Credit Notes sent by suppliers through the PEPPOL network and to send them to the customer via e-TrustEx.

More information about UBL documents is available in: https://www.oasis-open.org/committees/ubl/ .

The following diagram provides a short description of the context:
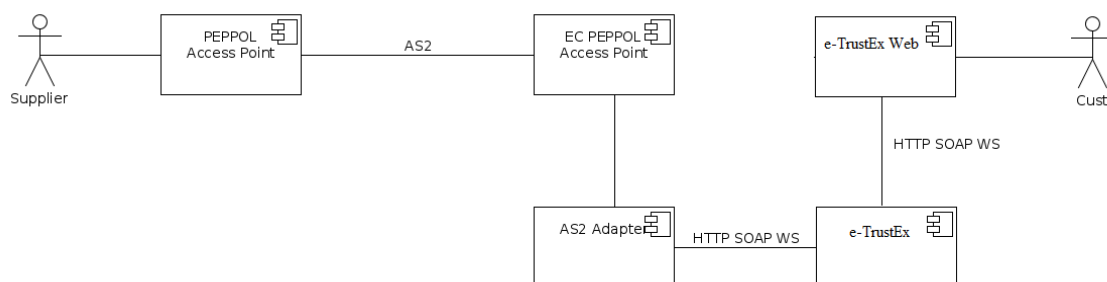
Fig. 2.   Architecture

A supplier sends via its PEPPOL Access Point an electronic document.

The Access Point transmits (using the AS2 protocol) the document to the Access Point included in this package, which is an instance of the Oxalis project (https://github.com/difi/oxalis) of the Norwegian Agency for Public Management and eGovernment.

The Adapter component in the above diagram retrieves the document and sends it to e-TrustEx that makes it available to the final recipient (the customer) via e-TrustEx Web.

3.1 Deployment diagram

The following diagram depicts the deployment model of the modules included in the current release:
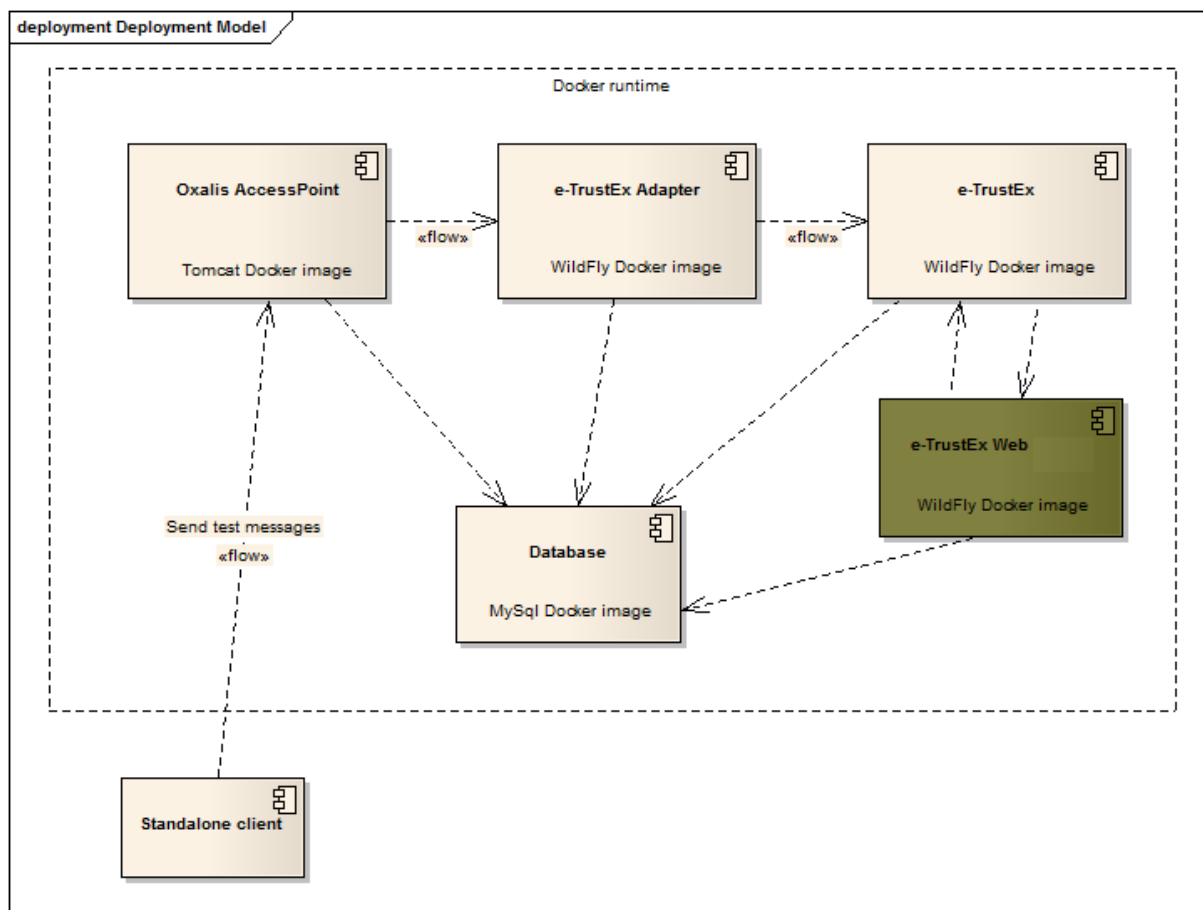
Fig. 3.   Deployment model

The modules of the system (Oxalis AccessPoint, e-TrustEx Adapter, e-TrustEx, e-TrustEx Web) are deployed as Docker containers based on customized Docker images.

Each of the four modules has its own DB schema configured inside the MySql Docker container.

More details about Docker can be found in the installation guide (chapter ***4. Installation guide***).

Outside the Docker context, the system also includes a standalone client (java application) that is used to simulate another PEPPOL AccessPoint that sends messages to the Oxalis AccessPoint in the above diagram.

3.2 Description of the modules

The four modules composing this package are described below.

3.2.1 Oxalis Access Point

The AccessPoint is a component of the PEPPOL network. General documentation about PEPPOL AccessPoints can be found on the PEPPOL website in the area "Access Point (AP) Implementation Guidelines" (https://peppol.eu/downloads/ap-guidelines/).

The Oxalis AccessPoint component in the above diagram is a customisation of the Oxalis open-source project (https://github.com/difi/oxalis) developed by the Norwegian Agency for Public Management and eGovernment.

It is customized on the level of message persistence: it stores into the database the messages that it receives from the PEPPOL network whereas in the default implementation of the Oxalis AccessPoint it writes those messages to a file store.

3.2.2 e-TrustEx

e-TrustEx is an open-source platform developed under the ISA programme. It is offered to Public Administrations at European, national and regional level to set up secure exchanges of digital structured and unstructured documents from system to system via standardised interfaces.

Complete details about the e-TrustEx platform can be found on the joinup page https://joinup.ec.europa.eu/software/openetrustex/description. The version used in this package is 'Open e-TrustEx 2.3.0' (https://joinup.ec.europa.eu/node/161366).

e-TrustEx supports different policy domains but in the context of the current document we are only using the e-Procurement domain. The services exposed by e-TrustEx are SOAP-based web services and the XML exchanged as payload respects the UBL syntax (https://www.oasis-open.org/committees/ubl/).

3.2.3 e-TrustEx Adapter

As its name suggests, this component plays the role of adapter between the message format exchanged in the PEPPOL network and the one supported by e-TrustEx.

Both platforms (PEPPOL and e-TrustEx) use the UBL syntax for the XML messages exchanged, but the difference lies in how the message is constructed.

In the PEPPOL world, the business message (Invoice or CreditNote) contains the attached documents embedded (as base64 content) into the XML.

e-TrustEx exposes separate services for the business documents (SubmitInvoice, SubmitCreditNote) and for the AttachedDocuments (SubmitAttachedDocument).

Therefore the Adapter processes the incoming message received from PEPPOL, it extracts any potential attached document embedded into it and calls the dedicated services exposed by e-TrustEx for the Invoice/CreditNote and the AttachedDocuments.

After the Invoice/CreditNote is stored e-TrustEx, its human-readable representation (PDF format) of the XML is generated asynchronously. Later the Adapter calls the SubmitViewRequest web service to retrieve the human-readable PDF from e-TrustEx, this service returns the base64 representation of the PDF.

The DocumentWrapper concept in e-TrustEx represents a document that will be part of a Document Bundle. The Adapter calls the StoreDocumentWrapper service in e-TrustEx for each of the following documents:

— XML of the Invoice/CreditNote

— human-readable PDF produced by e-TrustEx for the Invoice/CreditNote (as described above, the PDF is obtained by calling the SubmitViewRequest service)

— any document attached to the Invoice/CreditNote.

In the next step the Bundle XML, which contains metadata (DocumentWrapper ID, type, file name, size, hash method and hash) about all the DocumentWrappers, is built. Later on, this is used as payload of the SubmitDocumentBundle service.

Once created in the e-TrustEx database, the Bundle is routed to the e-TrustEx Web component to display its content to the receiver user, the Customer.

The current implementation of the Adapter assumes that the Sender and Receiver of the PEPPOL message are already configured in e-TrustEx, as well interchange agreements to send Invoice/CreditNote, AttachedDocument, DocumentWrapper and Bundle. More details about the configuration of the platform can be found in chapter **_5. Configuring the platform_**.

If the Sender/Receiver does not exist in e-TrustEx, the message will remain in a 'pending' state in the Adapter's database allowing it to be reprocessed after the necessary configuration has been done in e-TrustEx with the CIPAdmin tool (defined in **_5.2 Create new configuration using the CIPAdmin module_**).

3.2.4 e-TrustEx Web

e-TrustEx Web is an open-source module developed under the ISA programme. It is offered to Public Administrations at a European, national and regional level to view in a secure way the documents stored in e-TrustEx.

In the context of the current document e-TrustEx Web is configured with basic functionality to browse Invoices/CreditNotes by their recipient (see document "eInvoicing Viewer_Integration with PEPPOL – Quick User Guide").

3.3 Sequence diagram

The following sequence diagram provides a description of how the components interact.

Fig. 4.   Sequence diagram

When a Supplier sends a message through the PEPPOL network he uses his AS2 gateway that sends the message to the Oxalis AccessPoint instance. The AccessPoint saves the message in its database.

The new incoming messages are detected by the e-TrustEx Adapter component and a new workflow instance is started for each incoming message. The workflow is implemented using the Activiti BPMN workflow engine.

Activiti does the orchestration and calls different services to:

- retrieve and validate the message from the AccessPoint,

- resolve the Sender/Receiver parties,

- extract embedded attachments,

- send Invoice/CreditNote or Attached Document to e-TrustEx using web service calls,

- retrieve the Invoice/CreditNote human-readable PDF from e-TrustEx,

- store document wrappers of the Invoice/CreditNote XML and PDF, and any attached document sent by the Supplier together with the Invoice/CreditNote ,

- build the Bundle and send it to e-TrustEx using web service calls.

e-TrustEx stores the documents, the Bundles are routed to the e-TrustEx Web component to display its content to the receiver user (=the Customer) and eventually e-TrustEx notifies (push mechanism) the Customer that the documents are available or lets the Customer retrieve (pull mechanism) them as represented in the diagram.

More details about the two mechanisms can be found in chapter ***7. Creating a Customer e-Invoicing back-office***.

4

Installation guide

4 Installation guide

The platform uses Docker (https://www.docker.com/) for building and releasing the package that contains the Oxalis AccessPoint, the Adapter, e-TrustEx and e-TrustEx Web. It uses custom Docker images based on Tomcat, MySql 5.6 and WildFly 10.1.0.

The three images are preconfigured, meaning that at runtime:

- the MySql container contains the data model of the four applications

- the Tomcat container is preconfigured and runs the Oxalis AccessPoint application

- the WildFly container is also preconfigured (data sources, JMS queues, …) and the Adapter, e-TrustEx and e-TrustEx Web applications are deployed on it.

Both Tomcat and WildFly containers are linked at startup with the MySql container, in order for the applications to access the database.

The images are pushed as Public images on docker hub (https://hub.docker.com/).

4.1 Docker installation

Docker (and Docker Compose) should be installed according to the operating system used. After installation, run "*docker-compose version*" at the Docker command line to verify if Docker Compose is installed.

On specific Windows versions (eg: Windows 7 Enterprise), only Docker Toolbox can be installed.

If used behind a proxy, the following proxy settings should be written at the beginning of the file start.sh (found in the installation folder of Docker Toolbox):

- PROXY_USER=<XXX>

- PROXY_PASS=<XXX>

- PROXY_SERVER=<XXX>

- PROXY_PORT=<XXX>

- export http_proxy=http://${PROXY_USER}:${PROXY_PASS}@${PROXY_SERVER}:${PROXY_PORT}/ export https_proxy=${http_proxy}

- export HTTP_PROXY=${http_proxy}

- export HTTPS_PROXY=${http_proxy}

4.2 Running the application with Docker Compose

Compose is a tool for defining and running multi-container Docker applications (https://docs.docker.com/compose/).

It allows orchestrating different Docker containers and setting dependencies between them.

Assuming Docker Compose is installed, the following two files should be saved in a local folder:

- docker-compose.yml with the following content:
- mysql-peppol-viewer:
- environment:
- MYSQL_ROOT_PASSWORD: peppol
- image: digiteprocurement/mysql-peppol-viewer
- ports:
- - "3306:3306"
- volumes:
- - $MYSQL_DATA_VOLUME_PATH:/var/lib/mysql
- wildfly-peppol-viewer:
- links:
- - mysql-peppol-viewer
- ports:
- - "8080:8080"
- - "9990:9990"
- command: ["./wait-for-it.sh", "mysql-peppol-viewer:3306", "--", "/opt/jboss/wildfly/bin/standalone.sh", "-c", "standalone-custom.xml", "-b", "0.0.0.0", "-bmanagement", "0.0.0.0"]
- image: digiteprocurement/wildfly-peppol-viewer
- volumes:
- - $WILDFLY_PATH:/opt/jboss/wildfly/standalone/data
- tomcat-peppol-viewer:
- environment:
- OXALIS_HOME: /usr/local/tomcat/.oxalis/
- links:
- - mysql-peppol-viewer
- ports:
- - "8081:8080"
- - "443:443"
- image: digiteprocurement/tomcat-peppol-viewer
- volumes:
- - $OXALIS_KEYSTORE_PATH:/usr/local/tomcat/.oxalis/keystore
- .env with the following content:
- - MYSQL_DATA_VOLUME_PATH= /C/Users/vaicage/mysql-docker-volume
- - OXALIS_KEYSTORE_PATH= c:/Users/vaicage/oxalis-keystore
- - WILDFLY_PATH=c:/Users/vaicage/docker/wildfly-docker-volume

4.2.1 Updating the environment variables used by Docker

The above ".env" file should be edited to specify the real path on the local host for the two environment variables:

- MYSQL_DATA_VOLUME_PATH specifies the folder where the MySql Docker container should store the data on the local host of the user

- OXALIS_KEYSTORE_PATH specifies the folder on the local host of the user where the keystore containing the real PEPPOL certificate has been stored. This variable is used by the Tomcat container that runs the Oxalis AccessPoint. When the Tomcat container is started, this user-provided keystore will override the default one embedded into the Docker image (the one containing the self-signed certificate)

- WILDFLY_PATH specifies the folder where the WildFly Docker container should store the data from e-TrustEx (FILE_STORE_PATH parameter from etrustex.ETR_TB_METADATA table) on the local host of the user

If you are using Docker Machine on Mac or Windows, your Docker Engine daemon has only limited access to your macOS or Windows filesystem. Docker Machine tries to auto-share your /Users (macOS) or C:\Users (Windows) directory. More details here:

https://docs.docker.com/engine/tutorials/dockervolumes/#mount-a-host-directory-as-a-data-volume

Therefore it is advisable that the two environment variables should point to folders defined under /Users (for macOS) or C:\Users (for Windows).

4.2.2 Launching and shutting down the application

To **launch** the application:

1. From the docker command line (Docker Terminal) navigate to the folder where the above files (docker-compose.yml and .env) have been saved.

2. Once there, run the following command: "*docker-compose up -d*".

To **shut down** the containers:

1. Run "docker-compose stop && docker-compose rm -f"

4.3 Ports exposed by the Docker containers

- The MySql Docker image exposes the port 3306.

- The WildFly Docker image exposes the ports 8080 and 9990.

- The Tomcat Docker image exposes the port 8081.

4.4 Endpoints for accessing the applications of the platform

After the previous step is done, the applications should be up and running. For the first execution of the platform, it may take few minutes to complete the MySql container initialization. The DDL and DML scripts embedded in the Docker image are executed only at the first run with docker-compose. Subsequent executions will use the data saved by the MySql Docker container in the location specified above (the MYSQL_DATA_VOLUME_PATH, WILDFLY_PATH environment variable).

When the containers have finished initialization, the following URLs are available:

- http://localhost:9990/console is the admin console of the WildFly 10 application server.

  Login with user/password peppol/peppol.

Once connected, navigate to the Deployments tab and the three applications (etrustex.ear, etrustex-peppol-adapter.war and etx-wacc-modules-ear-wildfly.ear) should be "enabled" as in the screenshot below:
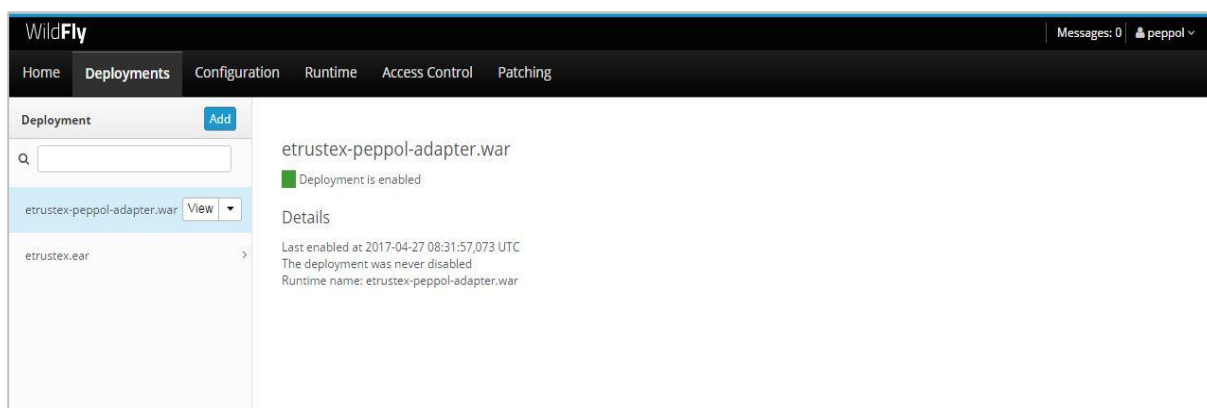
Fig. 5.   WildFly admin console

- http://localhost:8080/etrustex is the probing URL of e-TrustEx

  It shows the version number and the build date.

- http://localhost:8080/etrustex-admin-web/ is the CIPAdmin configuration tool.

  Login with user/password testadm/testadm.

  This module allows the technical configuration of the platform (Suppliers and Customers exchanging business documents). More details about this tool can be found in chapter **5.2 Create new configuration using the CIPAdmin module**
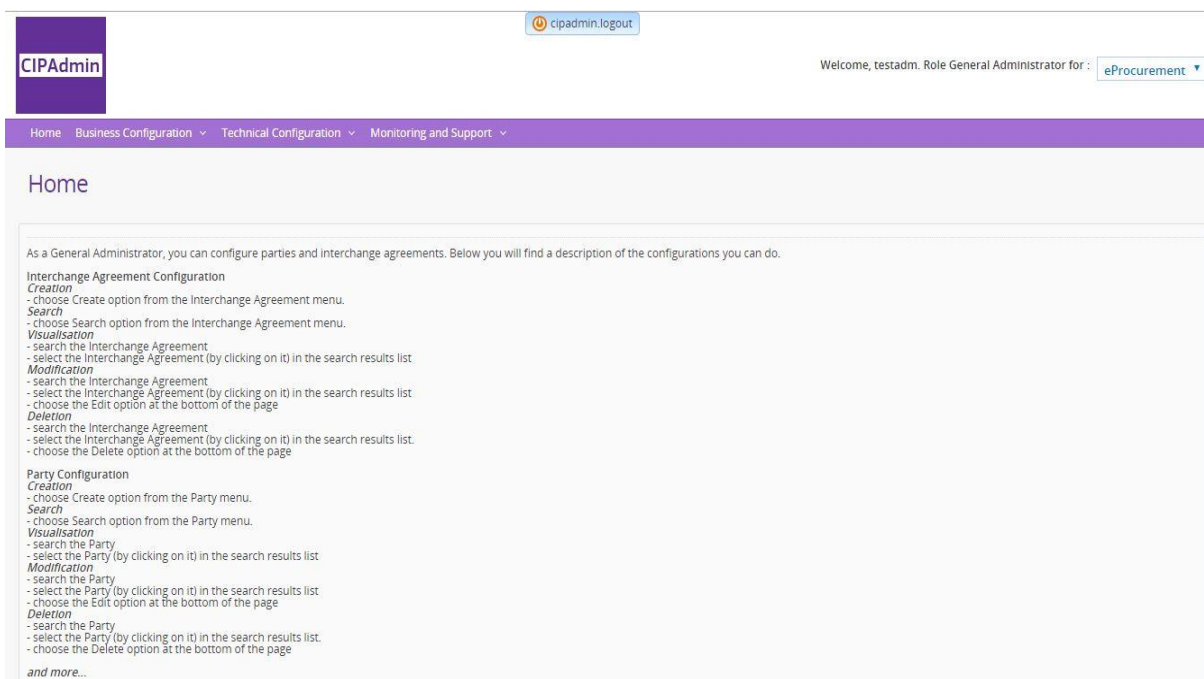
Fig. 6.   CIPAdmin console

- http://localhost:8080/ e-trustex/ is the e-TrustEx Web, a user interface that plays the role of a basic back-office system, allowing the Customer to see the business document sent by the Supplier.

  Login with user/password admin-web/admin-web.

  More details about this tool can be found in chapter **5.3 Create new configuration for the e-TrustEx Web module**

- http://localhost:8080/etrustex-peppol-adapter/login is the eTrustEx Adapter Administration Console.

  Login with user/pwd as2admin/as2admin.

- http://localhost:8081/manager/html is the admin console of Tomcat.

  Login with user/password manager/manager.

  It shows the status of the applications deployed and if successfully started, in the Applications table you should see the Path "/oxalis" with the status 'true' in the Running column.

Fig. 7.   Tomcat admin console

- http://localhost:8081/oxalis/as2 is the probing URL of the Oxalis AccessPoint running on Tomcat.

  It should return the message "Hello AS2 world".

4.5 Specific settings when running Docker Toolbox on Windows or Mac

At the time of writing this guide, Docker Toolbox is the only option available for using Docker on specific versions of Windows and Mac. The Toolbox installs Docker Client, Machine, Compose and Kitematic.

The following adaptations are needed for this particular configuration:

- 'localhost' in the URLs mentioned in the previous chapter (*3.4 Endpoints for accessing the applications of the platform*) should be replaced with the IP returned by the following docker command: "*docker-machine ip default*". In the default Docker Toolbox configuration, the IP returned is 192.168.99.100.

- the following sql statement should be run to update in the metadata table (OXA_ADAPTER_METADATA) the URL for calling the e-TrustEx web services (to replace 'localhost' with the IP returned by the previous step):

  - update peppol_adapter.OXA_ADAPTER_METADATA set MD_VALUE = 'http://192.168.99.100:8080/eprior/services/' where md_type = 'as2adapter_etrustex_endpoint';

  - update peppol_adapter.OXA_ADAPTER_METADATA set MD_VALUE = 'http://192.168.99.100:8080/etrustex/wrapperservices' where md_type = 'URL_STORE_BINARY_WEBSERIVCE';

  - update peppol_adapter.OXA_ADAPTER_METADATA set MD_VALUE = 'http://192.168.99.100:8080/etrustex/services' where md_type = 'URL_VIEW_WEBSERIVCE';

- update peppol_adapter.OXA_ADAPTER_METADATA set MD_VALUE = 'http://192.168.99.100:8080/etrustex/services' where md_type = 'URL_BUNDLE_WEBSERIVCE';

- the following sql statement should be run to update the URL for calling the e-TrustEx web services (to replace 'localhost' with the IP returned by the previous step) in the metadata table (ETX_WEB_CONFIG):

  - update etxweb.ETX_WEB_CONFIG set CFG_PROP_VALUE = 'http://192.168.99.100:8080/etrustex/services/DocumentWrapper-2.0' where CFG_PROP_NAME = 'etrustex.doc.wrapper.service.uri';

  - update etxweb.ETX_WEB_CONFIG set CFG_PROP_VALUE = 'http://192.168.99.100:8080/etrustex/services/DocumentBundle-2.0' where CFG_PROP_NAME = 'etrustex.doc.bundle.service.uri';

  - update etxweb.ETX_WEB_CONFIG set CFG_PROP_VALUE = 'http://192.168.99.100:8080/etrustex/services/RetrieveInterchangeAgreemen tsRequest-2.0' where CFG_PROP_NAME = 'etrustex.doc.agreement.service.uri';

  - update etxweb.ETX_WEB_CONFIG set CFG_PROP_VALUE = 'http://192.168.99.100:8080/etrustex/services/RetrieveRequest-2.0' where CFG_PROP_NAME = 'etx.node.services.RetrieveRequestService.uri';

  - update etxweb.ETX_WEB_CONFIG set CFG_PROP_VALUE = 'http://192.168.99.100:8080/etrustex/services/ApplicationResponse-2.0' where CFG_PROP_NAME = 'etx.node.services.application.response.url';

- the following sql statement should be run to update the URL for calling the e-TrustEx Web web services (to replace 'localhost' with the IP returned by the previous step) in the table (ETR_TB_ENDPOINT_WS):

  - update etrustex.ETR_TB_ENDPOINT_WS SET edp_ws_url='http://192.168.99.100:8080/e-trustex/integration/services/node/DocumentBundleService/v2.0' WHERE id='1';

  - update etrustex.ETR_TB_ENDPOINT_WS SET edp_ws_url='http://192.168.99.100:8080/e-TrustEx/integration/services/node/ApplicationResponseService/v2.0' WHERE id='2';

The MySql image exposes port 3306 and the password of the 'root' user is 'peppol'. A DB client application (like DbVisualizer) can be used to connect to the MySql Docker container and execute the update statement. More details in chapter *4.6 Connecting to the database*

- the default capacity settings (CPU/RAM) of the docker machine 'default' should be changed in the Oracle VM VirtualBox Manager GUI and the following steps should be followed:

1. First stop the VM from the command line (Docker Quickstart Terminal): run "*docker-machine stop*".

2. In Oracle VM VirtualBox Manager, go to Settings->System and change to at least 2 CPUs and 2048 RAM.

3. Afterwards restart the VM from the command line: run "*docker-machine start*".

4.6 Connecting to the database

A DB client application (like DbVisualizer) can be used to connect to the MySql Docker container with the following connection parameters:

- URL of the database server: *localhost* (or the IP from chapter **4.5 Specific settings when running Docker Toolbox on Windows or Mac**)

  - port: *3306*

  - database: *peppol_adapter*

  - user: *root*

  - password: *peppol*

4.7 eTrustEx WEB requirements

These are the current supported browsers and required settings for eTrustEx WEB:

Browser	Required	Additional information
Chrome	64	
Firefox	59	Requires these 2 preferences to be enabled in about:config. — javascript.option.streams — dom.streams.enabled
Edge	17	This is the EdgeHTML version. Edge is only available in Windows 10

5

Configuring the platform

5 Configuring the platform

5.1 Use the existing default configuration

The system is preconfigured with the following parties:

— Supplier 'TRUSTSUPPARTY1', configured in the system with the identifier 'TRUSTSUPPARTY1' of type 'GLN'

— Customer 'TRUSTCUSTPARTY1', configured in the system with the identifier 'TRUSTCUSTPARTY1' of type 'GLN'

You will find the above identifiers in the sample SoapUI project which is part of the current release.

See chapter *6.1 Using the provided SoapUI sample project* for more details.

The platform uses the notion of 'third-party user' to identify the system sending documents to the platform on behalf of the Supplier and to identify the back-office system of the Customer. The third-party user is identified by a username and a password to be used in the HTTP basic authentication part when calling the web services exposed by the platform.

In this default configuration, the Supplier is linked with the third-party 'AS2ADAPTER' identified by:

— username 'AS2ADAPTER', with the password 'AS2ADAPTER'.

The Customer is linked with the third-party 'SP1_USER' identified by:

— username 'SP1_USER', with the password 'SP1_USER'.

The above usernames/passwords are already configured in the sample SoapUI project.

Additionally, interchange agreements for the mentioned above Supplier and Customer, and profiles 'Toolbox', 'eInvoicing', 'Bundle' are predefined in e-TrustEx.

To dispatch Bundle documents from e-TrustEx to e-TrustEx Web the Web Services Routing Endpoints configuration is added in e-TrustEx.

5.2 Create new configuration using the CIPAdmin module

The CIPAdmin web tool (in order to access it, please refer to the link in chapter *4.4 Endpoints for accessing the applications of the platform*) is an admin console that allows provisioning the following:

- parties (Suppliers and Customers), including party identifiers

- interchange agreements: the technical link between the Supplier and Customer, authorizing them to exchange specific business documents with the platform

- routing endpoints (JMS or web services) for the back-office

- usernames (and corresponding passwords) to be used in the HTTP basic authentication when calling the web services of the platform

The user guide of this tool (*user_guide_cipa_adminconsole_open_etrustex.pdf*) is available here:

https://joinup.ec.europa.eu/software/openetrustex/document/open-e-trustex-230-documentation

5.3 Create new configuration for e-TrustEx Web module

The e-TrustEx Web module is configured at database level. The following elements can be managed:

- business configuration information,

- parties – specify customer parties to which documents user can have granted access,

- users – configure new users, grant roles, assign parties or business to which documents user can have access.

The user guide "e-TrustEx Web – Wildfly installation and configuration.doc" of this tool is available here.

Please refer to the chapter "3 WEB DATABASE CONFIGURATION" of the mentioned document.

6 Sending business messages to the platform

6 Sending business messages to the platform

There are two options available for the implementer to test the platform. They are described below.

The platform assumes that the Supplier and Customer used in the sample calls are already configured. See chapter **_5. Configuring the platform_** for more details on party configuration.

6.1 Using the provided SoapUI sample project

The release contains a sample SoapUI project (https://joinup.ec.europa.eu/svn/openeprior/branches/V_2_1_0_POST_AWARD/Tests/SoapUI/eInvoicing-2-1-soapui-project.xml) that allows calling the web services exposed by the platform. The project uses the preconfigured Sender/Receiver parties mentioned above in section **_5.1 Use the existing default configuration_**

It can be used to simulate both the Supplier and the Customer used in the e-Invoicing context.

6.1.1 As Supplier

For the Sender (Supplier) side it allows sending Invoices/CreditNotes and AttachedDocuments linked to them using the submitInvoice, submitCreditNote and submitAttachedDocument steps in the SoapUI project.

6.1.2 As Customer

For the Receiver (Customer) side, it allows to call the following services to consume the messages received from the Supplier:

submitInboxRequest: returns all the messages received by the Customer and not yet marked as 'Retrieved' after consuming them in the Customer's back-office. Once a message is consumed by the back-office, it should be marked as 'Retrieved' in the system, meaning that the next call to SubmitInboxRequest will not return this message anymore. In order to mark a message as 'Retrieved', the implementer of this solution should call the RetrieveRequest service with the RetrieveIndicator set to 'true' in the SOAP request, as shown in Fig. 8 below.
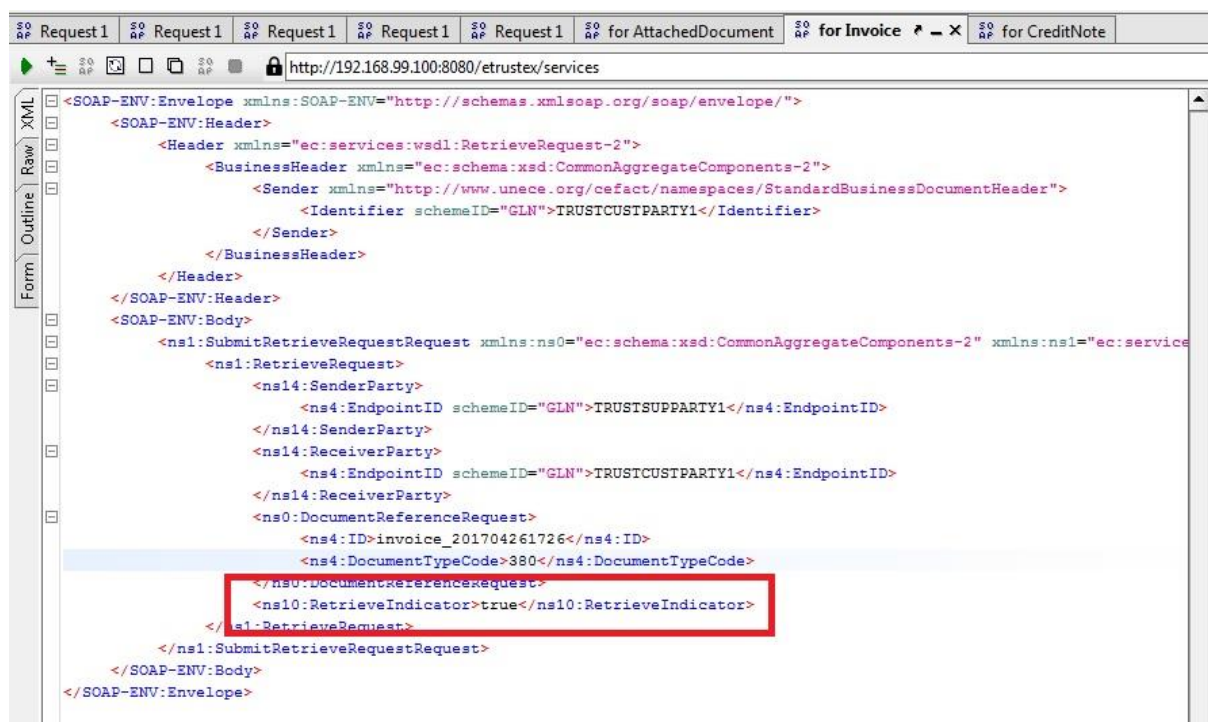
Fig. 8.   *Mark document as Retrieved*

- **submitRetrieveRequest**: it should be called for each business document returned by the InboxRequest service.

  - If it is called for an Invoice/CreditNote, it returns the UBL XML of the business document.

  - If it is called for an AttachedDocument it returns (in a HTTP multipart/related message) both the UBL XML of the AttachedDocument and the binary content of the AttachedDocument. Details below (Fig. 9) in a sample SoapUI response message:

Fig. 9.   RetrieveRequest for AttachedDocument

- **submitViewRequest**: it allows to obtain the PDF human-readable representation produced by the system for each Invoice or CreditNote received. The PDF is returned as base64-encoded binary in the SOAP response message.

- **submitApplicationResponse**: it is used to inform e-PRIOR about the result of processing the Invoice/CreditNote in the back-office. The payload of this SOAP request is an UBL XML message.

6.2 Using the standalone client that simulates a sending Oxalis AccessPoint

The standalone client is part of the official package released by the creators of the Oxalis AccessPoint (https://github.com/difi/oxalis). It is a java application that simulates a PEPPOL AccessPoint sending messages to the Oxalis AccessPoint which is part of the current solution.

It is important to note that the standalone client can only be used with a valid PEPPOL certificate. The default version included in this release uses a self-signed certificate that has to be replaced with a valid PEPPOL certificate obtained by the implementer of this solution.

6.2.1 Replacing the self-signed certificate with a valid PEPPOL certificate

So before being able to send any test message with this standalone client, the implementer of this solution should first obtain a valid PEPPOL certificate.

To obtain a valid PEPPOL certificate, follow the instructions provided in the chapter *__2.1 Obtaining a PEPPOL certificate for the AccessPoint.__*

6.2.2 Configuring the standalone client

1. Copy on your local drive the standalone client folder from the joinup SVN (https://joinup.ec.europa.eu/svn/openeprior/branches/V_2_1_0_POST_AWARD/Tests/standalone-client).

2. Once the certificate has been obtained and imported into a keystore file (keystore file should be named *oxalis-keystore.jks*), you should use it to override the default keystore file bundled in the client folder.

🗒️ **Password**

The default client configuration assumes that the password of the keystore is 'peppol'.

If a different password is selected, it should be set in the configuration settings, as explained below.

3. The following variables should be updated in the file *oxalis-global.properties*:
   - **oxalis.keystore** must contain the path where the client has been stored on the local host
   - **oxalis.keystore.password** contains the keystore password (if not the default one 'peppol')
   - **oxalis.inbound.log.config** and oxalis.app.log.config must contain the path where the client has been stored on the local host (pointing to logback-oxalis.xml)
   - **oxalis.operation.mode** contains TEST or PRODUCTION, depending on the certificate type.

4. Set the system variable OXALIS_HOME containing the path where the client has been stored on the local host.

6.2.3 Sample business messages provided in the package

This package contains two sample business documents (using UBL 2.1 syntax) for Invoice (https://joinup.ec.europa.eu/svn/openeprior/branches/V_2_1_0_POST_AWARD/Tests/standalone-client/BII04_v2_invoice1.xml) and CreditNote (https://joinup.ec.europa.eu/svn/openeprior/branches/V_2_1_0_POST_AWARD/Tests/standalone-client/CreditNote_PEPPOL_BIS_att.xml), both having embedded attached documents.

Please note that each request (Invoice/CreditNote) sent to the platform must contain a unique document ID for the specific Supplier/Customer pair used in the message.

The unique key imposed by the system is made up of: documentID + documentType (Invoice or CreditNote) + Sender (Supplier) + Receiver (Customer).

So in order to respect this unique key, the above sample files must be updated with unique IDs per request (Invoice/cbc:ID or CreditNote/cbc:ID).

6.2.4 Instructions on how to send messages to the Oxalis AccessPoint

Below is an example for running the client for sending a test message to the Oxalis AccessPoint included in this package:

java -Dmail.mime.foldtext=false -Doxalis.transmissionbuilder.override=true -jar oxalis-standalone.jar -f "C:/conf/oxalis/BII04_v2_invoice1.xml" -u http://localhost:8081/oxalis/as2 -m as2 -id APP_DIGIT_TEST -r 0088:TRUSTCUSTPARTY1 -s 0088:TRUSTSUPPARTY1

The parameters have the following meaning:

— **'-f'** specifies the absolute path to the test XML file. The path should be updated to your local path where the standalone client has been stored

— **'-u'** is the URL of the Oxalis AccessPoint instance

— **'-id'** specifies the identifier of the AccessPoint in the PEPPOL network. This value is the CN value from the PEPPOL certificate mentioned in chapter *5.2.1 Replacing the self-signed certificate with a valid PEPPOL certificate*. The current value used in this sample call (APP_DIGIT_TEST) is the CN value from the self-signed certificate provided by default in the client. As already mentioned, the self-signed certificate is to be replaced with a real PEPPOL certificate and its CN value should be used in the above call to the standalone client

— **'-s'** is the identifier of the Sender in the format schemeID:value

— **'-r'** is the identifier of the Receiver in the format schemeID:value

— The Sender and Receiver used in this call are the default Supplier/Customer described in chapter *5.1 Use the existing default configuration*

6.3 Checking the result of message processing in e-TrustEx

Once a message has been processed by the platform, the user can see the result of this process (document status, XML content, PDF human readable representation) either by querying the e-TrustEx database schema or by using the provided SoapUI project mentioned above.

6.3.1 Using the database

Follow the instructions in chapter *4.6 Connecting to the database*

Once connected you can run the following queries on the e-TrustEx schema:

- select * from etrustex.etr_tb_message where msg_document_id = 'AAA'; 'AAA' to be replaced with the actual document ID used in the sample XML message (value set in Invoice/cbc:ID or CreditNote/cbc:ID).
- select * from etrustex.etr_tb_message_binary where msg_bin_msg_id = MSG_ID;

Replace the MSG_ID parameter with the msg_id returned by the previous query.

In the result list of this query, the entry having msg_bin_type = 'RAW_MESSAGE' contains in the BLOB column 'msg_bin_file' the XML content of the business document sent by the supplier.

The entry having msg_bin_type = 'HUMAN_READABLE_MESSAGE' contains in the BLOB column 'msg_bin_file' the PDF human readable representation automatically generated by e-TrustEx for the Invoice and CreditNote.

6.3.2 Using SoapUI

The above mentioned sample SoapUI project can also be used to check the result of message processing in e-TrustEx.

The **'retrieveRequestBinding'** step can be used to call the RetrieveRequest service to obtain the XML content of the business document (Invoice or CreditNote) sent by the Supplier. If this service is called for an AttachedDocument, it returns both the XML and the binary content of the attachment.

The **'viewRequestBinding'** step can be used to call the ViewRequest service to obtain the PDF human-readable representation produced by e-TrustEx for each Invoice or CreditNote received. The PDF is returned as base64-encoded binary in the SOAP response message.

6.4 Business validation (Schematron)

The e-TrustEx component does a set of validations when receiving the business message for processing.

One of these validations is a business-rules validation implemented with the Schematron framework.

The current release uses the Schematron files released under PEPPOL Post-Award Business Interoperability Specifications (BIS), version BIS2.0-VA-V3.3.0. More details can be found here: https://peppol.eu/downloads/post-award/

7

Annex:

Creating a customer e-Invoicing back-office

7 Annex : Creating a customer e-Invoicing back-office

This chapter describes the two integration patterns offered by e-TrustEx for Customers willing to connect their e-Invoicing back-office to consume the business messages received from Suppliers. It explains how the messages received in e-TrustEx from the Suppliers can be consumed and how the status updates can be submitted to e-TrustEx once the back-office has consumed the business documents.

The term e-PRIOR used in the diagrams below refers to the eProcurement-related services exposed by the more generic e-TrustEx platform used in this package.

⚠ **IMPORTANT NOTICE**

The current package does not include any default Customer back-office implementation.

The information below provides guidance on the two options available for plugging a back-office system to the platform.

7.1 Store and Forward messaging pattern

This pattern is a mix of JMS and web services usage. It is described in the diagram below.
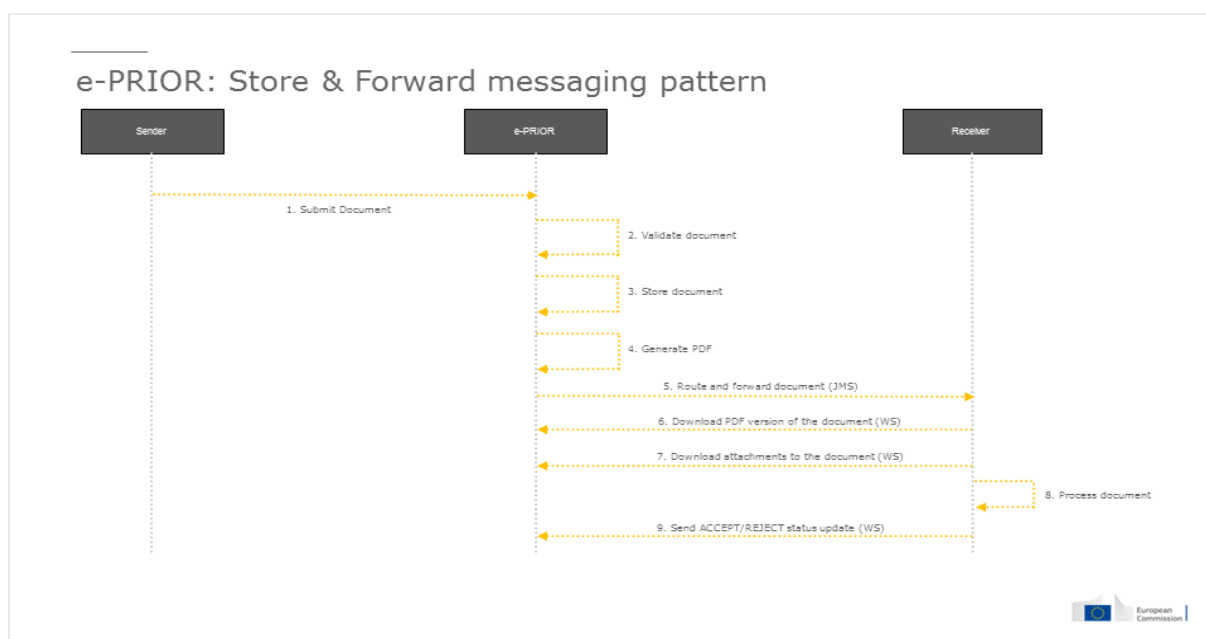

Fig. 10.   Store&Forward integration pattern

e-PRIOR is forwarding the business messages to a JMS queue in the customer's back-office.

In order to configure a JMS queue where the messages will be routed to, please refer to the CIPAdmin user guide, in chapter *5.2 Create new configuration using the CIPAdmin module*.

What is sent over JMS:

- the business documents metadata (UBL XML of Invoice and CreditNote)

- the metadata of AttachedDocuments (UBL XML). Not the binary content of the attachment.

For each invoice and credit note received from the suppliers, e-PRIOR generates a human readable version in PDF format. The binary content of the attachedDocuments sent by the supplier is also stored in e-PRIOR. Back-office systems can obtain both the human readable and the attachments' binary content by calling web services exposed by ePrior.

The customer system will have to call web services for the following:

- for each AttachedDocument XML received in the queue, call the RetrieveRequest service to obtain the binary content

- for each Invoice and CreditNote received in the queue, call the ViewRequest service to obtain the PDF human readable.

- once the processing of the invoice/creditNote has been done in the back-office, call the SubmitApplicationResponse service to update the status (ACCEPTED/REJECTED) of the business document

7.2 Store and Collect messaging pattern

This pattern uses only web services to pull information from e-PRIOR and to push status updates. It is described in the diagram below.
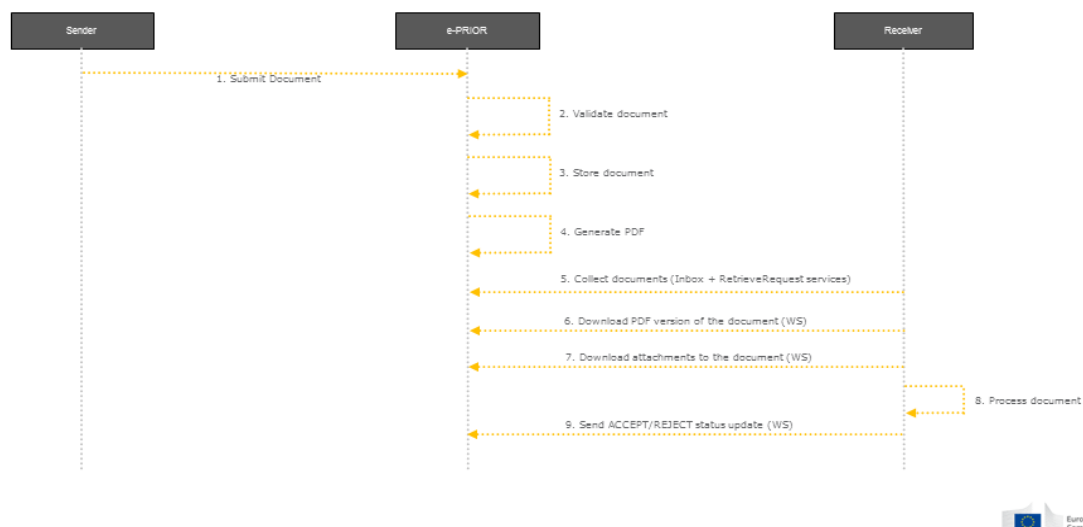
e-PRIOR: Store & Collect messaging pattern

Fig. 11.   Store&Collect integration pattern

The central point in this integration pattern is the Inbox concept used by e-PRIOR to store documents received by a specific party. In the context of this document, the Inbox stores the eInvoicing documents received by the customers.

When calling the InboxRequest service, the customer back-office receives the most recent 500 documents (Invoice/CreditNotes and their related AttachedDocuments) created in e-PRIOR for that customer and not yet marked as 'Retrieved' by the customer.

When processing the InboxResponse SOAP message, the customer system will have to:

- for each Invoice and CreditNote received, call the RetrieveRequest service to get the XML of the business document

- for each AttachedDocument found in the InboxResponse, call the RetrieveRequest service to get the XML metadata and the binary content

Once the processing of the Invoice/CreditNote has been done in the back-office, the next actions are mandatory:

- call the SubmitApplicationResponse service to update the status (ACCEPTED/REJECTED) of the business document

- call the RetrieveRequest service to mark the document as 'Retrieved' in e-PRIOR, in order to not appear anymore in the next calls to InboxRequest. This is an important aspect, linked to the limitation of 500 documents in the response, mentioned above. If the back-office does not mark documents as 'Retrieved' they will pile-up in e-PRIOR and once the limit of 500 documents is reached, the oldest ones will be ignored, so not returned in the InboxResponse.

7.3 Description of the web services

The web services exposed by e-TrustEx are SOAP based web services, using HTTP Basic Authentication.

The WSDLs of the web services are below:

– http://localhost:8080/eprior/wsdl/Invoice-2.1.wsdl

– http://localhost:8080/eprior/wsdl/CreditNote-2.1.wsdl

– http://localhost:8080/etrustex/wsdl/AttachedDocument-2.0.wsdl

– http://localhost:8080/etrustex/wsdl/InboxRequest-2.0.wsdl

– http://localhost:8080/etrustex/wsdl/RetrieveRequest-2.0.wsdl

– http://localhost:8080/etrustex/wsdl/ViewRequest-2.0.wsdl

– http://localhost:8080/etrustex/wsdl/ApplicationResponse-2.0.wsdl

– http://localhost:8080/etrustex/wsdl/DocumentWrapper-2.0.wsdl

– http://localhost:8080/etrustex/wsdl/DocumentBundle-2.0.wsdl

In case of DockerToolbox, please refer to chapter *4.5 Specific settings when running Docker Toolbox on Windows or Mac*, to replace 'localhost' in the URLs above with the real IP address of the docker-machine running on your computer.

SubmitInboxRequest: returns all the messages received by the Customer and not yet marked as 'Retrieved' after consuming them in the Customer's back-office. Once a message is consumed by the back-office, it should be marked as 'Retrieved' in the system, meaning that the next call to SubmitInboxRequest will not return this message anymore. In order to mark a message as 'Retrieved', the implementer of this solution should call the RetrieveRequest service with the RetrieveIndicator set to 'true' in the SOAP request.

SubmitRetrieveRequest: it should be called for each business document returned by the InboxRequest service (in the Store and Collect pattern) and for each AttachedDocument received in the JMS queue (in the Store and Forward pattern).

- If it is called for an Invoice/CreditNote, it returns the UBL XML of the business document.

- If it is called for an AttachedDocument it returns (in a HTTP multipart/related message) both the UBL XML of the AttachedDocument and the binary content of the AttachedDocument.

SubmitViewRequest: it allows obtaining the PDF human-readable representation produced by e-PRIOR for each Invoice or CreditNote received. The PDF is returned as base64-encoded binary in the SOAP response message.

SubmitApplicationResponse: it is used to inform e-PRIOR about the result of processing the Invoice/CreditNote in the back-office. The payload of this SOAP request is an UBL XML message.

The response can be either ACCEPT or REJECT. To differentiate between the two types, the ResponseCode element in the payload is used, with the following possible values:

- '380:1' for Invoice ACCEPTED
- '380:2' for Invoice REJECTED
- '81:1' for CreditNote ACCEPTED
- '81:2' for CreditNote REJECTED
- '916:1' for AttachedDocument ACCEPTED
- '916:2' for AttachedDocument REJECTED

The SoapUI project provided in this release (***6.1 Using the provided SoapUI sample project***) contains two sample request messages for ACCEPTED/REJECTED.