|  |  |
| --- | --- |
| logo_ec_17_colors_300dpi | EUROPEAN COMMISSION  DIRECTORATE-GENERAL INFORMATICS  Directorate D – Digital Services  **DIGIT D3 – Trans-European Services** |

**Architecture analysis and impact assessment**

**for eTrustEx Web**

Date: 22/02/2019

Version: 1.00

Author(s): Pablo GUERRERO-GOMEZ

Reviewer(s): Kelly LILJEMO, Kristof POPGEORGIJEV

Approver(s):

**Contents**

[1. Introduction 3](#_Toc534940764)

[2. Available options 3](#_Toc534940765)

[2.1 Option 1. Baseline. Using ETX-NODE for all communications (AS IS) 3](#_Toc534940766)

[2.2 Option 2. Using ETX-NODE only for GUI – System. Not for GUI – GUI 5](#_Toc534940767)

[2.3 Option 3. Using eDelivery AP 7](#_Toc534940768)

[2.3.1 Option 3a. Via Domibus AP. 4 Corner Model for internal and external systems 8](#_Toc534940769)

[2.3.2 Option 3b. Via Domibus AP. 4 Corner Model for external and 3 Corner model for internal 8](#_Toc534940770)

[2.3.3 Option 3c. Web interface as part of AP plugin 9](#_Toc534940771)

[2.4 Option 4. Using AP for external systems and direct for internal 10](#_Toc534940772)

[2.5 Option 5. With plugins for AP and via ETX-NODE 11](#_Toc534940773)

[3. Preferred option 12](#_Toc534940774)

[4. Conclusion 13](#_Toc534940775)

[Annexes 14](#_Toc534940776)

[Annex 1. ETX-NODE services 14](#_Toc534940777)

[**Annex 2. eDelivery AP services** 15](#_Toc534940778)

[Annex 3. File size limitations tests 16](#_Toc534940779)

***Definitions, acronyms and abbreviations***

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| *AP* | *eDelivery's Domibus Access Point* |
| *BE Plugin* | *Backend connector* |
| *ETX-NODE* | *eTrustEx Node* |
| *ETX-WEB* | *eTrustEx Web* |
| *FE Plugin* | *Frontend connector* |
| *GUI* | *Graphical User Interface, eTrustEx GUI* |
| *Node Plugin* | *Node connector* |
| *Plugin* | *AP Connector* |
| *WS* | *Webservice* |

# Introduction

**The context**

There is an increasing demand to use ETX-WEB in different domains, followed by an increasing number of requests for customizations.

Currently, ETX-WEB relies on the ETX-NODE for exchanging documents, both among GUI users and Back-end systems.

ETX-NODE and other Systems exchange messages using Adapters, which will be soon replaced by eDelivery APs.

**Problem definition**

The current ETX-WEB application suffers from important legacy issues such as;

- hard and expensive to maintain.

- difficulty to integrate with other systems,

- poor User Experience

An analysis of its architecture and GUI is relevant and topical.

**Objectives**

The aim of this document is to analyse, from ETX-WEB point of view, its optimal fit in the new ecosystem along with ETX-NODE and eDelivery APs. How ETX-WEB maintenance, performance and usability can be improved.

# Available options

## Option 1. Baseline. Using ETX-NODE for all communications (AS IS)

This is the baseline from which the options are assessed.

This scenario depicts the "GUI - system" and the "GUI - GUI" exchange of messages using ETX-WEB.

- The interaction between ETX-WEB and ETX-NODE as it is today.

- The interaction between ETX-NODE and systems as it will be when existing Adapters are replaced by APs.

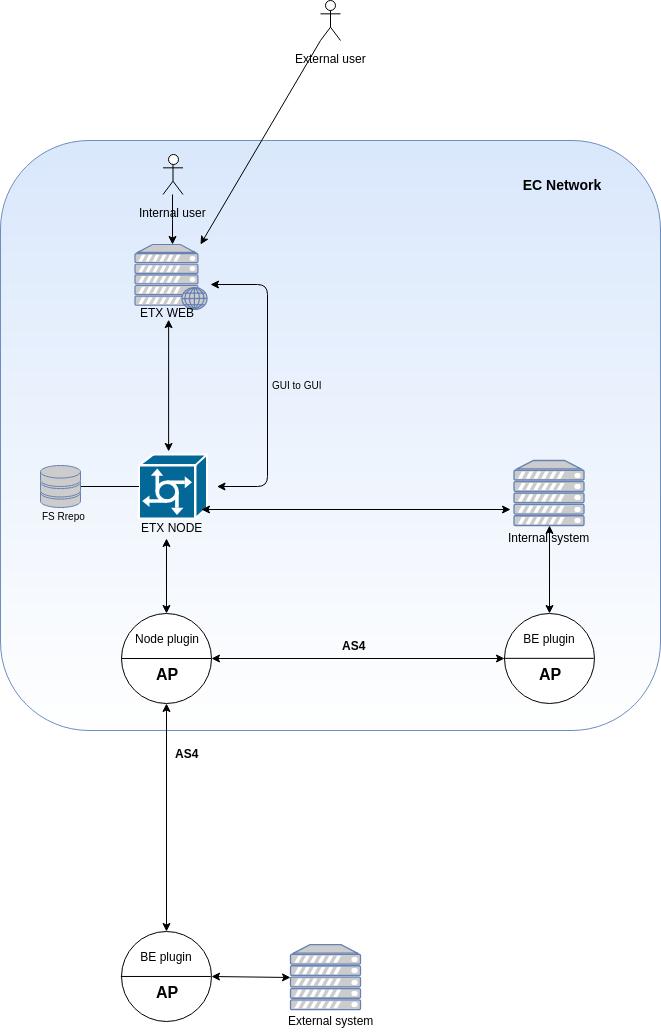
Communication between ETX-NODE and Internal systems can be either direct or via APs.

ETX-NODE Plugin translates between Domibus default plugin specification and ETX-NODE's custom specification (Bundle & Wrapper).

Currently, ETX-WEB relies on ETX-NODE for all transmissions. Detailed information about this interaction can be found in <https://webgate.ec.europa.eu/CITnet/confluence/x/dxJ2Lw>.

Impact:

* It makes ETX-WEB dependent on ETX-NODE data model.
* It requires duplicated configurations on ETX-NODE & ETX-WEB databases (Parties, ICAs…) and handling synchronization between them.
* It imposes ETX-NODE file size limitations.
* There is a transfer time overhead between ETX-WEB and ETX-NODE.
* ETX-WEB must either implement ETX-NODE custom specification (DocumentBundle, DocumentWrapper, …), or do a similar job as the NODE plugin does and translate between specifications.
* There will be redundant services. Most services provided by ETX-NODE, of which ETX-WEB benefits are already provided by APs.



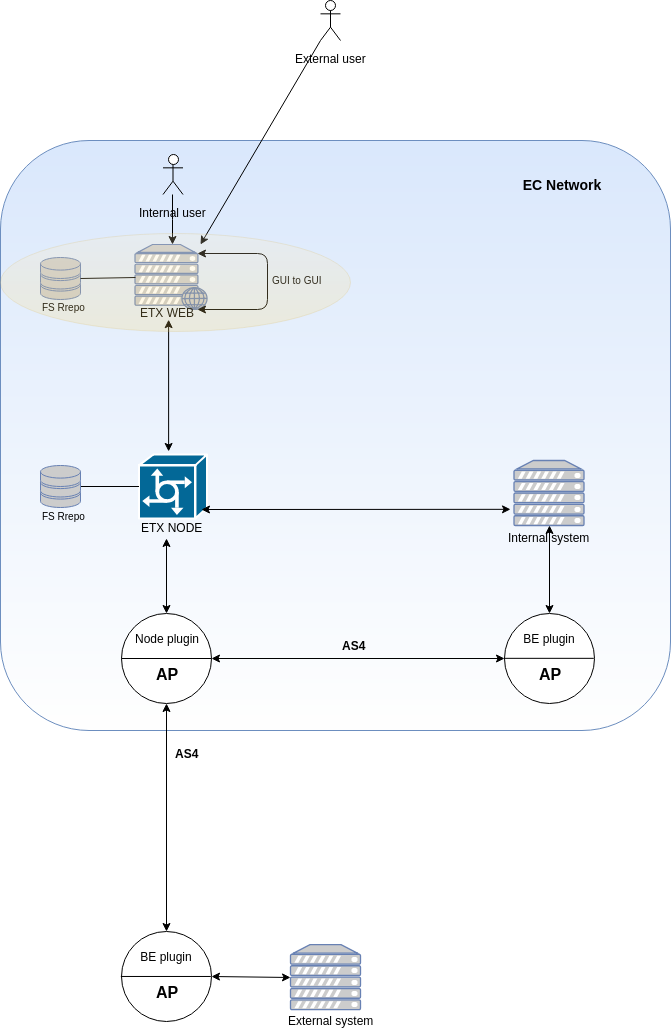
## Option 2. Using ETX-NODE only for GUI – System. Not for GUI – GUI

In this scenario, GUI to GUI messages are entirely handled by ETX-WEB, which needs its own file repository.

Here, GUI to GUI messages are not impacted by the architecture complexity issues listed in previous scenario.

GUI to GUI messages do not need all the WS related services offered by ETX-NODE, but of course, all the business logic related to storing in its own file system needs to be implemented.

If messages arriving to ETX-WEB will be handled differently depending on whether the receiver is a system or a human, this distinction has to be configured and handled.



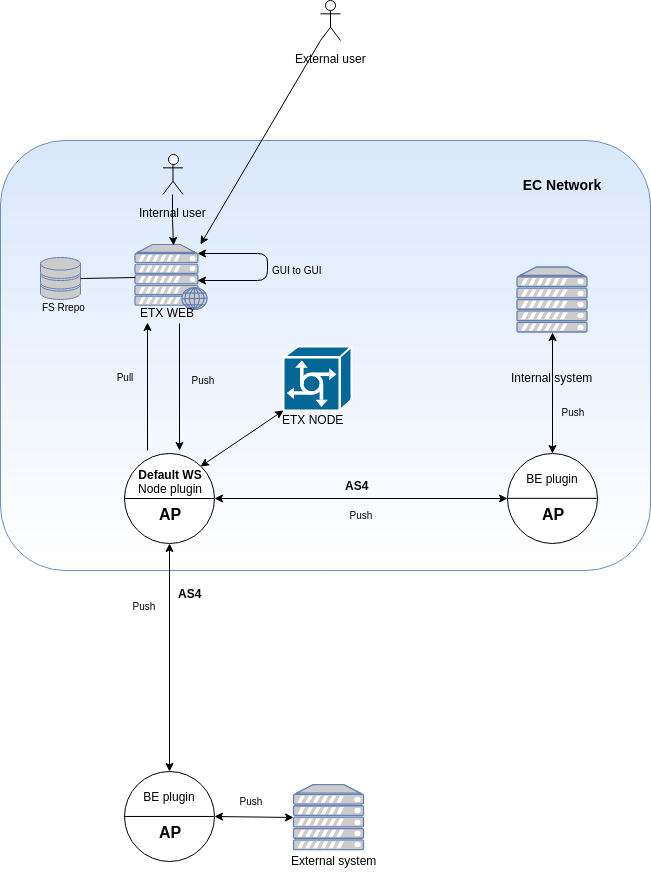
## Option 3. Using eDelivery AP

The following two scenarios have the same impact on ETX-WEB. Both show how ETX-WEB can exchange messages with systems using APs directly (without ETX-NODE).

Impact:

* There is no dependency on ETX-NODE data model.
* There is no need for duplicated configurations on ETX-NODE & ETX-WEB databases (Parties, ICAs…) and handling synchronization.
* There are no file size limitations imposed by ETX-NODE. By using APs, there is a 2 Gb size limit per transmission.
* It is possible to use Domibus default plugin WS specification directly. No need for translation. ETX-WEB can implement an interface for the default WS specification exposed by ETX-NODE plugin, or continue using ETX-NODE specification and NODE’s plugin.
* No redundant services along the processing of messages.

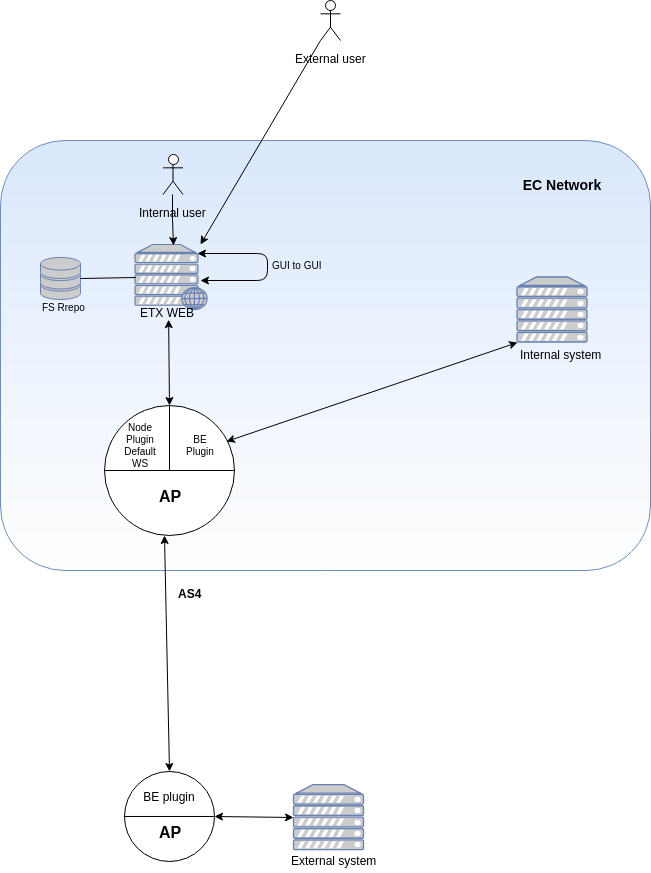
### Option 3a. Via Domibus AP. 4 Corner Model for internal and external systems



### Option 3b. Via Domibus AP. 4 Corner Model for external and 3 Corner model for internal

A variation of the previous scenario for the case in which system hosted inside the EC network will use an AP with both, FE and BE plugins.

In principle, ETX-WEB is not impacted by this change. The interface exposed by the AP would be the same, just the endpoint of the AP plugin would change for ETX-WEB.

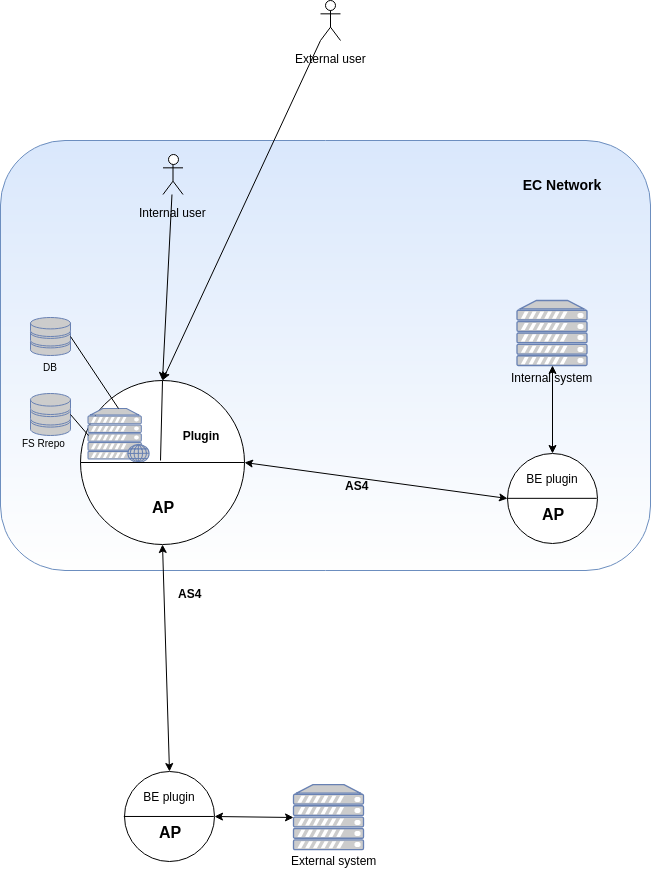


### Option 3c. Web interface as part of AP plugin

ETX-WEB disappears and instead a GUI is developed as part of the AP FE plugin.

In this scenario, all the ETX-WEB functionalities would have to be implemented as part of the plugin.

The plugin would need its own file repository and database.



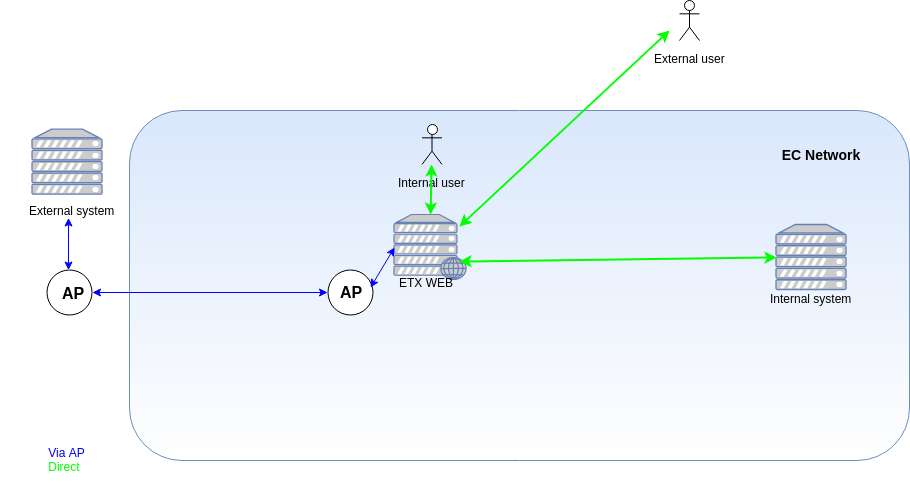
## Option 4. Using AP for external systems and direct for internal

ETX-WEB already does E2E encryption, signing and integrity validation. Thus, https protocol should be sufficient for GUI - SYSTEM communication (as it is for GUI - GUI). Although direct GUI – EXTERNAL SYSTEM exchange is technically feasible, in the context of the EU SEND architecture, communications with external systems is planned via eDelivery APs.

For GUI – INTERNAL SYSTEM exchange ETX-WEB has to expose SOAP webservices, REST, GraphQL or other. A Web client jar can be published on nexus to make it easier for clients to exchange with ETX-WEB via its REST API.

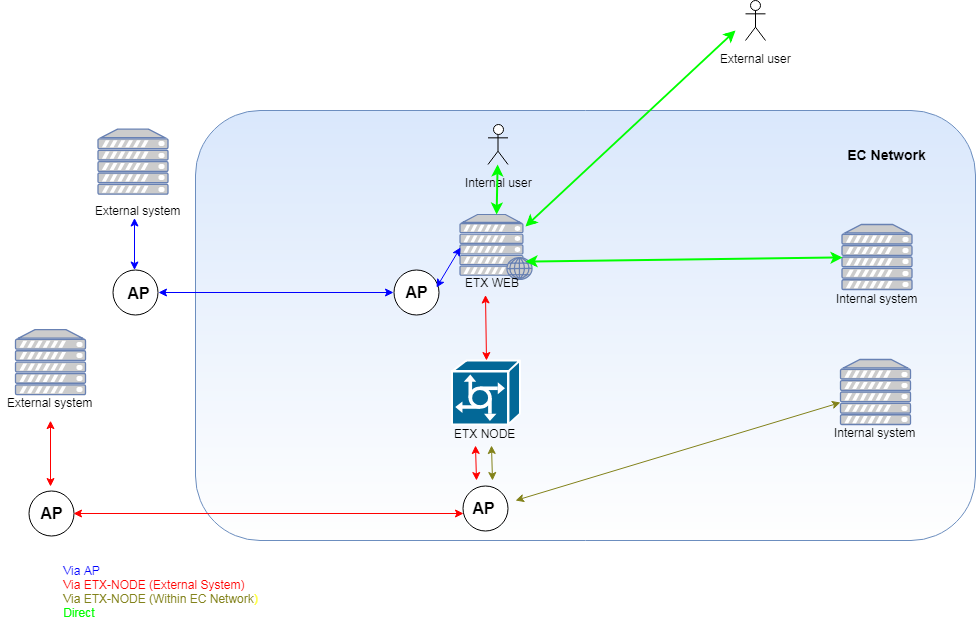
For GUI – EXTERNAL SYSTEM exchange, at the time of writing, it is not yet decided how the ETX-WEB will connect to the eDelivery AP (several plugins for such connections are available).

The implementation effort required on ETX-WEB side is the same as for any other system using eDelivery AP.



## Option 5. With plugins for AP and via ETX-NODE

In this variation of the previous scenario, ETX-WEB can also exchange messages via ETX-NODE to avoid impact on clients that are already using ETX-NODE.



**NOTE**: ETX-WEB exchange with internal systems could also use AP if required but for simplicity sake it is not shown in the diagram.

# Preferred option

Option 1 (AS IS) is unnecessarily complex. There are duplicated services provided by both, ETX-NODE and APs. ETX-WEB is highly coupled to ETX-NODE and several services provided by the latter are not needed by the former.

Option 2 (Standalone only for GUI – GUI) does not solve the negative issues from previous option. It only reduces them to the GUI – System communication, but with a high implementation effort. It is not a cost-effective solution.

Option 3 (Via AP) may suit situations in which APs are required. There is no difference between 3a or 3b options from ETX-WEB point of view, but this scenario introduces a new dependency on the eDelivery AP for System – GUI communication, a high implementation effort for modifying the AP plugins and increases the complexity of the overall architecture. Furthermore, coupling the ETX-WEB with APs may not suit all the requirements of all customer, hence less flexible than option 5.

Option 3c (Web interface as part of AP plugin) came up during discussion but it was discarded as an option because they are two different applications, with different scopes and release lifecycles. It would introduce undesired coupling on both sides. For example, every time a new, patch, bug fix, change request, etc. for the GUI has to be implemented and deployed, the AP plugin would have to be re-deployed as well and vice versa.

Option 4 (Using AP for external systems and direct for internal) simplifies exchange with internal systems and uses AP just for exchange with external systems. This option however does not take into account the high implementation effort required from legacy systems that are currently using the System – GUI communication via ETX-NODE, including a possible risk of duplicating ETX-NODE functionalities in the AP and/or the ETX-WEB.

Option 5 (With plugins for AP and via ETX-NODE) is the preferred option. It is the best compromise between simplicity and fulfilling client requirements. Furthermore, it gives the most flexibility in terms of all other options for using the new ETX-WEB in the context of EU SEND. By using plugins, ETX-WEB is decoupled from other systems, thus becomes a more maintainable, standalone building block that can be reused in many scenarios without the dependencies existing with the current ETX-WEB.

# Conclusion

The new ETX-WEB should be a standalone building block, that is able to provide E2E encryption, signing and verification of content and signatures for file exchange among GUI users and backend systems.

Its modular design makes the overall architecture maintainable, as well as provide flexibility when it comes to reusing it as a building block in multiple scenarios.

In the current context, the new ETX-WEB should be able to exchange files via APs or ETX-NODE if required. Therefore, the development of the new version of ETX-WEB should be based on Option 5.

**Considerations regarding file size limitations**

* ETX-WEB standalone
  + GUI Upload: unlimited size with modern browsers
  + GUI Download: unlimited size only with Chrome and 512MB with other browsers (until [Streams specification](https://streams.spec.whatwg.org/) is completed)
  + System Upload/Download: unlimited size
* ETX-WEB via AP: 2Gb per transmission.
* ETX-WEB via ETX-NODE: 1Gb per transmission.

# Annexes

## Annex 1. ETX-NODE services

#### Schematron validation - Redundant

This is specific for e-procurement and so far, not requested by ETX-WEB users.

#### Message correlation - Redundant

There are no conversations or messages replies in ETX-WEB.

#### Advanced logging - Redundant

In ETX-NODE, logging is associated to the thread across the stack of service activators. In ETX-WEB there is no such integration; logging can be easily implemented with aspects or other mechanism.

#### Retention policies - Redundant

ETX-WEB needs its own specific retention policies in any case.

#### Message ID validation (length, content, duplication) - Redundant

ETX-WEB needs implementing its own in any case.

#### Data extraction from XML - Redundant

Used by e-Justice. Not required by ETX-WEB users.

#### Support for MTOM - Redundant

ETX-WEB implements (or should) in any case.

#### Re-dispatching - Redundant

From the GUI, messages can be re-dispatched manually (new CR from DG-COMP)

#### Retries - Redundant if using APs

Provided by AP.

#### XSD validation - Redundant if using APs

Extra XSD validations for asynchronous processing of messages. The default connector of the AP already provides synchronous validation and asynchronous validation (for certificates and transactions results, by pulling),

#### Admin console

Currently, ETX-WEB relies on ETX-NODE's ICA's, Parties and Endpoint configurations. For that, ETX-NODE's admin console is used.

**Annex 2. eDelivery AP services**

* Non-repudiation
* Retries
* AS4 Standard
* Dealing with downtimes

## Annex 3. File size limitations tests

#### Introduction

eTrustEx Web team has performed a series of tests and POCs to study existing and potential file size limitations.

#### Context

The context of these tests is a full Javascript application that provides end to end encryption, signing, signature verification and integrity validation.

#### Results table

|  |  |
| --- | --- |
| Webgate (from outside EC network) | **1.4 GB** (file in single request)  **Unlimited** (file in chunks) |
| eTrusteEx WEB Current version | Depends on the user’s RAM availability and browser. After tests on different machines/browsers, we concluded that **512 MB** is safe |
| eTrustEx NODE | **1 GB** |
| eTrusteEx WEB UPLOAD (in chunks) | **Unlimited** |
| eTrusteEx WEB DOWNLOAD (using ServiceWorker & MessageChannel to save files. C**urrently only available in Chrome**) | **Unlimited** |

#### Conclusions

Web client-side development is evolving towards stream programming models.

Server side, Servlet 3.1 non-blocking I/O API allows full handling of request and response as streams.

Client side, browsers are slowly implementing the Streams API. We cannot yet process the Request body as a stream when uploading a file, but we can overcome that by uploading in chunks.

However, for download, although we can process the Response Body as stream in modern browsers, only Chrome currently allows us to pipe the processed stream into a file saved in the client’s drive.

Webgate tests uploading files from outside the EC network failed at a bit more than 1GB. This limit can be bypassed by handling files in chunks, both for upload and download.

**Limits within what may be expected for a web application (we stopped tests at 20GB):**

- Upload: any size with any browser (excluding IE).

- Download: any size with Chrome and 512MB with other browsers (excluding IE).