

**COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME****ICT Policy Support Programme (ICT PSP)**

ICT PSP call identifier: ICT PSP-2008-2

ICT PSP main Theme identifier: CIP-ICT-PSP.2008.1.1

Project acronym: SPOCS

Project full title: Simple Procedures Online for Cross-border Services

Grant agreement no.: 238935

Technical Overview of WP3 Interoperability Framework

Deliverable Id :	D3.3
Deliverable Name :	Open Modules
Status :	Corrigenda 2.0
Dissemination Level :	SPOCS internal and EU Commission
Due date of deliverable :	28-2-2011
Actual submission date :	24-03-2011
Work Package :	WP3 Interoperable delivery, eSafe, secure and interoperable exchanges and acknowledgement of receipt
Organisation name of lead contractor for this deliverable :	BVA (DE)
Author(s):	bremen online services (DE), BVA (DE), Siemens SIS (DE), InfoCert (IT)
Partner(s) contributing :	SPOCS AT (AT)

Abstract: This is the third deliverable in work package 3 of the EU co-funded project SPOCS. It describes the open modules implemented as specified with the foregoing deliverable. This revision 2.0 is a formal adjustment of the sections related to eDelivery, realized with implementation version 1.3.0.



History

Version	Date	Modification reason	Modified by
0.1	05.01.2011	Initial draft	bos
0.1.1	07.01.2011	Appendix A – Acceptance criteria	BVA, bos
0.2	7.1.2011	Draft of content/ section headers	Jörg Apitzsch
	18.1.2011	Revision Structure & Appendix A	
	20.1.2011	Revision again	
	28.2.2011	Structure as agreed in call	
0.3	7.2.2011	Section2 deleted (New) Section 2 draft from CS	Jörg Apitzsch Christian Schmitt
0.31	8.2.2011	Section 3.3 Input Bernd (included by Jörg, some stuff moved to 3.2 (common) + Update of D3.2 eSafe	Bernd Martin Jörg Apitzsch
0.4	10.2.2011	Contributions Siemens merged	Bernd Martin Christian Schmitt Jörg Apitzsch
0.5	10.2.2011	Merged sections not considered in 0.4	Jörg Apitzsch
0.51	11.2.2011	Section 1 Introduction (parts) Section 1.3 and 1.4 Appendix B	Stefanie Rieger
0.6	16.2.2011	Deliverable ready for internal review	Jörg Apitzsch
0.7 / 0.7.1	03.03.2011	Deliverable ready for external review	Bernd Martin Jörg Apitzsch Lars Thölken Luca Boldrin



1.0	23.03.2011	External Review comments implemented, ready for submission to the European Commission	Lars Thölken
1.0.1	09.05.2011	Add disclaimer watermark	Lars Thölken
1.1		Deliverable approved by the European Commission	
2.0	26.03.2012	eDelivery adjustments with respect to latest revision of ETSI TS 102 640 and implementation version 1.3.0. Subsection pointing to LT's UI-based Test Gateways included.	Jörg Apitzsch Wilko Oley

Pending EC Approval



Table of Contents

History	2
Table of Contents.....	4
List of Figures	6
List of Tables	7
List of Abbreviations.....	8
Executive Summary	10
1. Introduction to the Deliverable.....	12
1.1. Corrigenda of underlying Specification D3.2	13
1.2. Relations to internal SPOCS Environment.....	13
1.3. Relations to external SPOCS Environment.....	13
1.4. Quality Management.....	14
1.5. Risk Management.....	16
2. Security Model - Results of Security Architecture Development Process and Formal Modeling & Verification	16
2.1. Initial Situation	16
2.2. Result of Formal Modelling and Verification.....	16
2.3. Need for Action.....	17
3. Open Modules Development, Operational Platforms and Testing	18
3.1. Common.....	19
3.2. eDelivery Specialities.....	20
3.2.1. Contents of the Development.....	20
3.2.2. Domestic eDelivery Integration to SPOCS Interconnect.....	20
3.3. eSafe Specialities	22
3.3.1. Contents of the Development.....	22
3.3.2. Recommended Integration Process of the Modules	22
3.3.3. Integration of the Modules and Implementation of the SPOCS Process	24
4. Technical Overview on Open Modules.....	26
4.1. SPOCS TSL	26
4.1.1. Basic Elements	26
4.1.2. TSLinterface usage.....	31
4.2. Generic eDelivery Gateway	33
4.2.1. Packages Overview	33
4.2.2. Package messages.....	34
4.2.3. Package messageparts.....	34
4.2.4. Further Objects	35



4.2.5.	Hints to implement Adapters to domestic eDelivery solution	35
4.2.6.	Testing.....	37
4.3.	Generic eSafe Connector Modules	40
4.3.1.	eSafe Open Modules Release Notes	40
4.3.2.	eSafe Open Modules Delivery Items	40
4.3.3.	eSafe Open Modules technical Architecture.....	42
4.3.4.	Overview on the Procedure for Integration	42
4.3.5.	Testing the eSafe Open Modules	44
5.	Conclusions	50
	References	51
A.	Appendix A – Acceptance Criteria	52
B.	Appendix B – Risk List.....	55
C.	Appendix C – Formal Modelling and Verification Results.....	60
D.	Appendix D – TSL Interface Code Usage Example	67



List of Figures

Figure 1: SPOCS Developers Wiki – General Environment Info Page	18
Figure 2: eDelivery components in context of domestic MD's	21
Figure 3: Overview of SPOCS scenario within a Test-environment	23
Figure 4: Class TSL	26
Figure 5: Class TSLEntry	28
Figure 6: Class TrustedProviderImpl	29
Figure 7: Class TrustedServiceImpl	30
Figure 8: Hudson build process monitoring	38
Figure 9: eSafe Open Modules technical architecture	42
Figure 10: Demo Portal Home Page	45
Figure 11: Demo Portal About Demo page	46
Figure 12: Demo Portal Document Transfer Status page	47
Figure 13: Demo Portal Transfer Finished page	48
Figure 14: Demo Stress Test Demo page	49



List of Tables

Table 1: eDelivery Open Modules packages	34
Table 2: eDelivery Open Modules tested platforms	37
Table 3: eSafe Open Modules tested platforms	40
Table 4: eSafe Open Modules delivery items	41
Table 5: Acceptance criteria list	53
Table 6: Risk List	58

Pending EC Approval



List of Abbreviations

Abbreviation	Explanation
API	Application Programming Interface
CA	Competent Authority
DB	Database
DTP	Document Transfer Package
EC	European Commission
eID	Electronic Identity
ETSI	European Telecommunications Standards Institute
EUPL	European Union Public Licence
GW	(eDelivery) Gateway
IdP	Identity Provider
IETF	Internet Engineering Task Force
IP	Internet Protocol
JAXB	Java API for XML Binding
JEE	Java Enterprise Edition
JSF	Java Server Faces
LSP	Large Scale Pilot
MD	(eDelivery) Management Domain
MS	Member State
NTP	Network Time Protocol
OASIS	Organization for the Advancement of Structured Information Standards
OCD	Omnifarious Container for eDocuments
OCL	OCD Container lightweight
OCF	OEBPS Container Format
OSOR	Open Source Observatory and Repository
PEPPOL	Pan-European Public Procurement Online
PSC	Point of Single Contact
PTC	Technical Coordinator (of the SPOCS LSP, in this case)
REM	Registered E-Mail
SD	Services Directive
SP	Service Provider
SPI	Service Provider (WS) Interface



SPOCS	Simple Procedures Online for Cross-border Services
STORK	Secure Identity Across Borders Linked
SVN	Apache Subversion
TSL	Trust-service Status List
TSP	Trusted Service Provider
UDDI	Universal Description, Discovery and Integration
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WP<n>	Work Package (number n)
WS	Web Service
WSDL	Web Service Description Language
WS-I	WS-Interoperability Organization
XML	Extensible Markup Language

Further abbreviations used in this document are explained on first occurrence.



Executive Summary

SPOCS – Simple Procedures Online for Cross-border Services – is aimed at providing seamless cross-border electronic procedures for setting up a business in another EU country, in the context of the Services Directive. The project builds on solutions developed in Member States as they implement the Services Directive. Building on compliancy with the Services Directive, SPOCS is also about competitiveness and it has been set up on the basis of the 2008 CIP ICT PSP Programme of Work (project reference: 238935).

As described in the SPOCS Technical Annex, the current deliverable is the result of the task “Task T3.4: Implementation of required modules and/or gateway functionality”.

D3.3 describes the common technical platform for functionality and interfaces of open software modules to (i) enable cross-border interoperability of different eDelivery and eSafe systems and (ii) to implement a secure interoperable connection for data and document exchange between the heterogeneous infrastructures of different countries; thus meeting the functional and security requirements of each solution in a fully interconnected way.

eDelivery focuses on the path of data transmission between two communication partners. Those can be human endpoints with their user agents or even applications acting on message data in an automated way. eSafes, on the other hand, are systems designed to archive, manage and share data as a source for “authenticated documents”. Both systems are required to support processes within the context of the Service Directive and are active components within the SPOCS scenario. Both are also required to fulfill maximum security within their internal operations but also for the communication with others.

The implemented modules consider and incorporate existing national infrastructures, by combining their positive aspects, thereby gaining the benefits of results in similar fields/topics of sibling LSP projects, such as STORK (Pan European Proxy Services - PEPS) and PEPPOL (Business Document Exchange Network- BusDox and e-Signature Validation Infrastructure).

While implementing the technical specifications from D3.2 [1], a strong motivation was to support acceptance beyond the project borders and to ensure long term sustainability. In order to propose an integrated solution, suitable for SPOCS business cases, highly accepted and state-of-the-art standards have been incorporated wherever needed. These are specifications provided by ETSI (Registered E-Mail – REM, Trust-service Status List - TSL), as well as several others from W3C, IETF, WS-I, and OASIS.

While implementing the open modules, some specialties arose, which are mentioned and described in sections 3 and 4 of this deliverable and of course solved within the open modules.

During development, minor changes were made for parts of the deliverable D3.2 with respect to the eSafe specification. Therefore some revised documents from the previous deliverable are added to the current deliverable. Specifically some chapters of the main document as well as the appendix 4 with operations in detail and the eSafe WSDL files were updated. The OCD example is not included anymore since it is part of the WP2 deliverable D2.3.

This document provides an overview and general guidelines for implementing connectors for domestic eDelivery and integrating eSafe solutions already in place to interoperate via the open modules provided. All modules are implemented in the platform-independent Java programming language, thus gaining maximum technology neutrality regarding operational environments. For detailed open modules functionality documentation, standard Javadoc is provided, available



online in the a common “Reference Environment” established by the project. This Environment has been set up for all SPOCS technical work packages, details of which are described in the “SPOCS-Wiki” as main entry point: https://dev.spocs.bos-asp.eu/wiki/index.php5/Main_Page.

Open Modules, released under the EUPL v1.1, together with their online documentation, are the core part of this deliverable, which is planned to be made available in the EC repository “Joinup”¹ in May 2012.

During the piloting phase ahead, the open modules are available for partners and developers within the LSP SPOCS via the Reference Environment, which uses modern, free available tools to support the collaborative development and maintenance process (for further details see section 3).

The next phase will focus on the support of the piloting countries. It is the major goal to successfully deploy the modules developed in the SPOCS piloting countries, with the ability to solve the addressed interoperability issues. The modules must successfully be adopted and integrated in various implementations. If needs of further improvement of the modules will be visible during the national implementation phase and even piloting one, this will lead to well defined iteration steps of according enhancements.

¹ <https://joinup.ec.europa.eu/>



1. Introduction to the Deliverable

The project: SPOCS

The Large Scale Pilot 'Simple Procedures Online for Cross-Border Services' (SPOCS) deals with enhancing cross-border communication between Service Providers (SP) and Points of Single Contact (PSC) by electronic means, to support the implementation of Article 8 of the Services Directive. One of the major objectives of SPOCS is to enable and support procedures, in accordance with the service directive, with the focus on interoperable and secure data exchange between Member States.

Work Package 3: Scope and objectives

Regarding the objectives for Work Package 3, we have to point out two topics: eDelivery and eSafe and additionally the overlaying Security Architecture.

To reach the objective for **eDelivery** the intention of WP3 is to implement a SPOCS enabled process to provide an eDelivery channel from the PSC's Member State to the Member State of the Service Provider (i.e. receiver). Thus, the major challenge for WP3 eDelivery was to achieve a technical concept enabling cross-border eDelivery by using the domestic eDelivery channels of the Service Provider's Member State. The solution had to be found while taking into consideration the fact the Service Provider (i.e. receiver) should not be required to create an additional eDelivery account in the country of the PSC. *Instead, the Service Provider's existing home system should be used for e-Delivery.*

Thus eDelivery focuses on the path of data transmission between two communication partners. These can be human end-points with their client computers or protected services like eSafes.

Within the potential scenarios (use cases), the **eSafe** building block is another alternative option. Providing easy and harmonised authorised and authenticated access to the private contents of eSafes across Member States is considered essential. With this vision in mind, it is essential to have a common understanding of an eSafe.

In general, an electronic data and document safe (eSafe) is a virtual repository for storing, administering and sharing personal electronic data and documents. It provides storage of, and access to archived documents for authorised parties in a secure manner, makes online transactions more efficient, convenient and user friendly and provides a source of authentic documents for eGovernment processes.

WP3 integrates eSafes with the objective of sharing documents, selected and authorised by the owner. This should confirm and prove that using eSafes is an efficient, secure and easy way to exchange relevant documents (e.g. certificates) between SP and PSC. The implementation of the required interfaces and the required functionalities for the document exchange between eSafes and PSCs is the task of this deliverable.

Security architecture is a set of representations that describe the function, structure and interrelationship of the security components within an environment.

Each component - eDelivery and eSafe - is required to fulfil maximum security requirements within its operation, and also for the communication with others.

This deliverable D3.3 document contains the overall description how to develop open modules and provides a structural overview including interfaces and dependencies and provides further necessary information for each component.



1.1. Corrigenda of underlying Specification D3.2

Due to perceptions made in the implementations phase, minor parts of the specifications provided with D3.2 had to be detailed and/or modified. The Open Modules provided implement these updated version of the specifications.

The following documents of D3.2 are affected:

- SPOCS D3.2 Functional Specification, Architecture and Trust Model
- Appendix 2: Trust-service Status List Profiling ("SPOCS TSL")
- Appendix 3: eDelivery Interconnect Protocol and Gateway Specification
- Appendix 4: eSafe – Operations in Detail

The corrigenda versions v1.1+ and v2.0 will be published in the SPOCS portal² at the time of approval.

1.2. Relations to internal SPOCS Environment

Basically, dependencies between SPOCS WP's have not changed with regard to the foregoing specification phase, thus the according section of D3.2 [1] still applies.

Specific for the implementations phase is the following:

- All technical WP's had to agree on a common environment for development, test and deployment. For this purpose, a Reference Environment has been set up and standards where defined for the development process.
- Beyond mentioned common base, eDelivery open modules have no dependencies on software provided by other WP's.
- The SPOCS TSL implementation done by WP3 has to cover the requirements of WP1 and WP4 – as specified already in D3.2. According feedback on feasibility of SPOCS TSL open modules was requested and given.
- eSafe open modules use implementation results of WP2 (OCD), what required close mutual alignment.
- At least, all WP3 open modules will have to prove their viability and quality in the piloting phase ahead. WP3 is aware on reacting accordingly on feedback from WP5.

1.3. Relations to external SPOCS Environment

The WP3 concerns eSecurity and interoperable connectivity of eDelivery and eSafe services. This subject has been addressed by many efforts conducted in the last decade by IT industry, standardisation bodies and public institutions, especially for e-government and e-business.

The WP3 interoperability framework aims to interconnect distributed different solutions builds on open available implementations of the relevant specifications and profilings of W3C, IETF, OASIS and the WS-I organisation.

On European level, relevant ETSI specifications apply; here, ongoing awareness must be taken

² Most recent versions of SPOCS deliverables are available at: http://www.eu-spocs.eu/index.php?option=com_processes&task=showProcess&id=18&Itemid=61



to refinement respective reshaping of e-signature standards started in 2009. Efforts conducted by Member states and the EC for interoperability and convergence of national solutions are considered, concepts or solutions as mentioned in WP3 D3.2 [1] are adopted as far as fitting to the requirements as identified in WP3 D3.1 .

For eDelivery, concepts and solutions required from SPOCS WP3 must be go beyond to the concrete business scenarios served (content agnostic, at least). Thus, a generic approach has been chosen, feasible for e-government scenarios with comparable requirements. With the goal to obtain a broader, generic and broadly accepted approach towards an European eDelivery/e-Transport interoperability framework, there is an ongoing collaboration with the sister Large Scale Pilot projects. Especially ETSI's Specialists Task Force (STF 402) must be mentioned here, which in parallel to the SPOCS specification phase was working on "REM Interchange: e-mail Interchange between Registered E-Mail (REM) systems based on different transmission protocols"³. The REM SOAP Binding Profile of this ETSI specification TS 102 640 directly adopts the WP3 specifications on eDelivery.

Since there are no broad initiatives ongoing and no implementations for eSafes are available yet, the approach chosen was to reuse and adopt foreign domain concepts wherever possible. This led to the decision to use the OCD container (see WP2) as a transport container for the document exchange and to use the TSL approach as a basis for the trust relationship between communication partners.

1.4. Quality Management

To make sure the outcome of Task 3.4 „open modules“ is usable and fulfils its purpose, we follow in principle the dedicated quality process implemented for Deliverable D3.1, which of course, observes the WP7 Quality and Risk plan. In this phase, these quality actions focus mainly on the development process. The aim is to make sure the open modules correctly fulfil the specification of the Deliverable D3.2. Thus the open modules support the overall processes of the SPOCS big picture. Some special constructive and analytical quality actions follow:

Constructive quality actions

- Assignment of the main parts of development (eDelivery Gateway, SPOCS TSL, eSafe modules) to the team leaders of developers of the WP3 core partners (FHB, Siemens, Austria and Italy). When necessary, experts for specialist issues have been included.
- Creation of a schedule agreed by the team. For alignment with the other technical Work Packages and the piloting countries, this time-table was published on the internal SPOCS WP7 site.
- To produce this Deliverable 3.3 document a content list was created and agreed. Additionally, the progress of development was managed and supervised via a production list (title, responsibility of contribution and deadline).
- To ensure a seamless connection between modules and national connectors in time, the WP3/WP4 leader pushed to bring the first technical alignment meeting forward. Participants involved have been responsible developers and leaders of Work Packages 1 to 4, the piloting countries and overall piloting organisations from WP5 and WP7). To control the usability, a second alignment meeting followed barely 4 months later.

³ ETSI STF Homepage <http://portal.etsi.org/stfs/process/home.asp> (last visited on Sep. 11, 2010)



- Crucial for the success of a development (e.g. making sure reliability of results and the usage will be as expected and described before) is the preparation of adequate testing with the following rules and actions:
 - Every developing partner is responsible for preparing the needed test cases (in connection with the specifications from D3.2) as well as the development and testing, both functional and technical, equal to state-of-the art software production. For early tests, to ensure the smooth connection with other modules, the Reference Environment (provided by FHB) could simply be used.
 - Obligatory for all partners: to follow the given structure and interface rules, in addition to programme standards stipulated beforehand.
 - A testing environment (a special part of the Reference Environment with regulated and controlled access) was also provided by FHB to enable the tests to be run:
 - Automatic and controlled run of test cases, including history.
 - Test documentation, which includes a deficiencies list, to be produced per test object.
 - Change- and Configuration Management (including Version Management) is set up.
 - The complete environment, usage and rules are described in the wiki of the Reference Environment (available online at https://dev.spocs.bos-asp.eu/wiki/index.php5/Main_Page)
- The progress of work is regularly controlled on the basis of the production list, in phone calls/net meeting sessions (WP3 partners, PTC from WP7, supervised by WP3 leader)
- In the final phase, the core team is responsible for completing the provided parts and ensuring consistency.
- Collaboration meetings with other Large Scale Pilots (LSP) and programmes to ensure usefulness and capability of possible solutions.

Analytical quality actions:

- Deliverable 3.3 document:
 - Each updated version of the deliverable (document) is being provided to the WP3 partners and PTC for content approval (eDelivery, eSafe and Security Architecture)
 - The QA has been carried out internally by WP3:
 - The core team revised and aligned the deliverable and completed the links and layout.
 - The work package leader always supervised the process and carried out the final, formal check of D3.3. This deliverable then passed through a multi-stage quality SPOCS internal process. The next stage is the final check by the External Quality manager and the Programme Director, before its submission to the EC.
- Open modules
 - Testing took place in two steps:
 - Module Test
 - (Technical) Integration

The test cases and test documentation can be found in the SPOCS Subversion



repository. For further detailed information see section 4.2.6 (eDelivery) and 4.3.5 (eSafe)

1.5. Risk Management

Suitable Risk Management assists in achieving the objectives on time and with sufficient quality. Again, in principle the dedicated risk management defined is applied for Deliverable D3.3, which naturally observes the WP7 Quality and Risk plan.

The types of the probable risks influence our methodology and there is a strong connection between risk management and quality actions which are typically utilised to avoid risks. Especially constructive quality actions are quality management measures aimed to avoid risks, such as narrow time-span).

Coordinated with the subtask responsible people, the work package leader presents in each Executive Board an adjusted current risk list. Then the challenges and issues among all work packages can be managed together.

The risk list was regularly updated by WP-Leader and input of the partners was also considered.

The risks regarding the deliverable D3.3, and the risks concerning the whole WP3, had to be constantly under the work package leader's control and have been finally accepted during the quality phase of this deliverable.

2. Security Model - Results of Security Architecture Development Process and Formal Modeling & Verification

2.1. Initial Situation

Modern distributed systems are not monolithic designs but typically follow the Service Oriented Architecture (SOA) approach. The composition of existing security-sensitive services in a SOA-based infrastructure entails a wide range of trust and security issues.

Therefore SPOCS WP3 has empowered a security architecture development process (documented in "The SPOCS D3.2 Appendix1 Security Architecture Development Process") and developed a Security Model in order to illustrate security-related concepts such as the information assets processed, the roles/actors involved, the states of the system and the messages exchanged (documented in "The SPOCS D3.2 Appendix6 Security Model").

On the basis of this security model a formal analysis was conducted in order to reveal risks and weaknesses of the SPOCS security architecture.

2.2. Result of Formal Modelling and Verification

The formal modelling and verification of the SPOCS scenario and Open Modules has revealed a number of potential security problems within the SPOCS Infrastructure, which are caused by impersonation and Denial of Service attacks.

The following risks have been revealed

- Risks due to a TSL Denial of Service Attack
- Risks due to missing Registration and Authentication at PSCs



- Risks due to weak Registration and Authentication at PSCs
- Risks due to unverifiable Authentication of SPOCS Components against SP

As a consequence several security goals such as confidentiality, authenticity / integrity, and availability (as defined in Deliverable D3.2) are violated. The detailed description of the attack scenarios including proposed mitigation measures can be found in Appendix C – Formal Modelling and Verification Results”.

It is important to note that all security issues revealed are not rooted in the SPOCS interoperability framework but are nevertheless crucial for the overall security and therefore need to be addressed at a higher level than WP3. Efforts and measures to mitigate the risks pointed out need to be taken and coordinated at a European level.

No action taken by a single member state or WP3 alone can successfully address those threats. The resulting risk mitigation measures have to be implemented by the respective member states.

2.3. Need for Action

Before SPOCS, the impact of risks was locally isolated as there was no or limited interconnectivity between member-state specific components or solutions.

Since by interconnecting member-state specific components solutions, local risks will become risks at European level, the revealed Risks⁴ have to be:

- Analyzed and prioritized according to their potential impact and the respective threat profile – *Risk Assessment*
- Countermeasures and possible compensating security controls must be developed and implemented – *Risk Treatment*
- Residual risks that can not be mitigated must be documented and accepted – *Risk Treatment*

⁴ According to ISO/IEC 27001



3. Open Modules Development, Operational Platforms and Testing

This section summarizes the development- test- and maintenance tools as well as the runtime environment as agreed upon by all technical work packages, amended by specific details for eDelivery and eSafe.

For all SPOCS technical work packages, a common “Reference Environment” has been set up, details of which are described in the SPOCS Wiki: <https://dev.spocs.bos-asp.eu/wiki/index.php5/GeneralEnvironment>:

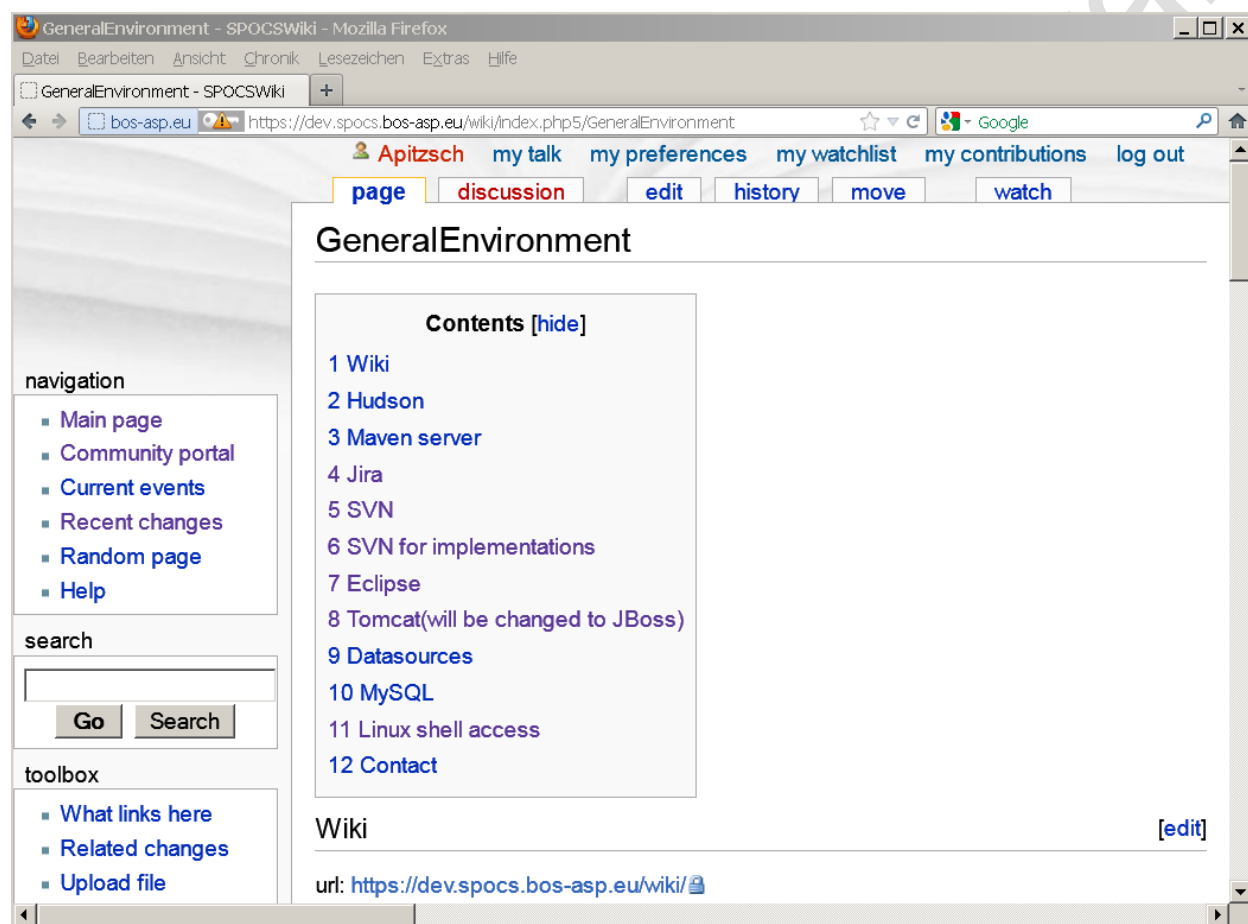


Figure 1: SPOCS Developers Wiki – General Environment Info Page

The Wiki is permanently updated on base of actual perceptions; all members of the SPOCS technical work packages as well as people concerned with technical issues during SPOCS piloting should consider this Wiki regularly and publish experiences here, as far rated useful for all partners concerned.



3.1. Common

Common development toolkits and runtime platforms used:

- **Subversion**

Apache Subversion⁵ is used as shared repository and version control systems in the Reference Environment: <https://svnnext.bos-bremen.de/SPOCS/AllWpImplementation/>

- **Java™**

Programming language for the main modules, whereas JUnit was used for the unit and component tests during the development (Classes, methods, functionalities are tested via JUnit tests in local development environment). For the current implementation phase as for the piloting one, JDK/JRE 1.6 is in use.

- **Jira**

Atlassian Jira⁶ is used for issue tracking bug reporting and development/maintenance processes monitoring: <https://dev.spocs.bos-asp.eu/jira/secure/Dashboard.jspa>

- **Tomcat/JBoss**

JBoss⁷ is used as the (reference) Application Server in order to provide the runtime environment for the modules. The modules should be deployable on other Application Servers as well.

WP3 Open Modules will not use specific J2EE-features. Thus, modules will be deployable on the Apache Tomcat JSP and Servlet Container⁸ too.

- **Eclipse**

Eclipse⁹ is used as the integrated development environment (IDE) for the major development of the modules.

- **Maven**

In order to support the build process Apache Maven¹⁰ is used as software project management and comprehension tool. A Sonatype NexusTM server¹¹ has been set up as Maven repository manager in the Reference Environment: <https://dev.spocs.bos-asp.eu/nexus/index.html#welcome>

- **Hudson**

Hudson is used as continuous integration server¹². Builds of new releases of SPOCS Open Modules and their results can be found [at https://dev.spocs.bos-asp.eu/hudson/](https://dev.spocs.bos-asp.eu/hudson/).

⁵ <http://subversion.apache.org/>

⁶ <http://www.atlassian.com/software/jira/>

⁷ <http://www.jboss.org/jbossas>

⁸ <http://tomcat.apache.org/>

⁹ <http://www.eclipse.org/>

¹⁰ <http://maven.apache.org/>

¹¹ <http://nexus.sonatype.org/>

¹² <http://hudson-ci.org/>



3.2. eDelivery Specialities

The eDelivery Open Modules are provided with the focus to support the integration of domestic eDelivery infrastructures to the SPOCS eDelivery interconnect network, using the trust establishment functionality via SPOCS TSL as specified in D3.2 [1].

3.2.1. Contents of the Development

Main parts provided are

- The “generic” eDelivery Gateway functionality, dealing with Gateway to Gateway communication on the basis of the Interconnect Protocol as specified in D3.2
- A convenience API, implementing the interfaces to adapters to domestic eDelivery solutions. This API must be used to implement the domestic (mapping) part of the eDelivery Gateway to be set up for each national eDelivery solution.
- A generic Gateway instance, deployed in the Reference Environment. This Gateway is intended to be used for simulating cross-border (respective cross-solution) message exchange when developing and testing a domestic Gateway. This generic Gateway instance provides no mapping functionality to an example or “template” domestic solution; it operates only on message formats as defined by the SPOCS Interconnect Protocol.

3.2.2. Domestic eDelivery Integration to SPOCS Interconnect

The following figure provides a brief overview of the scenarios which is necessary for testing and running the domestic eDelivery adapters to be set up:

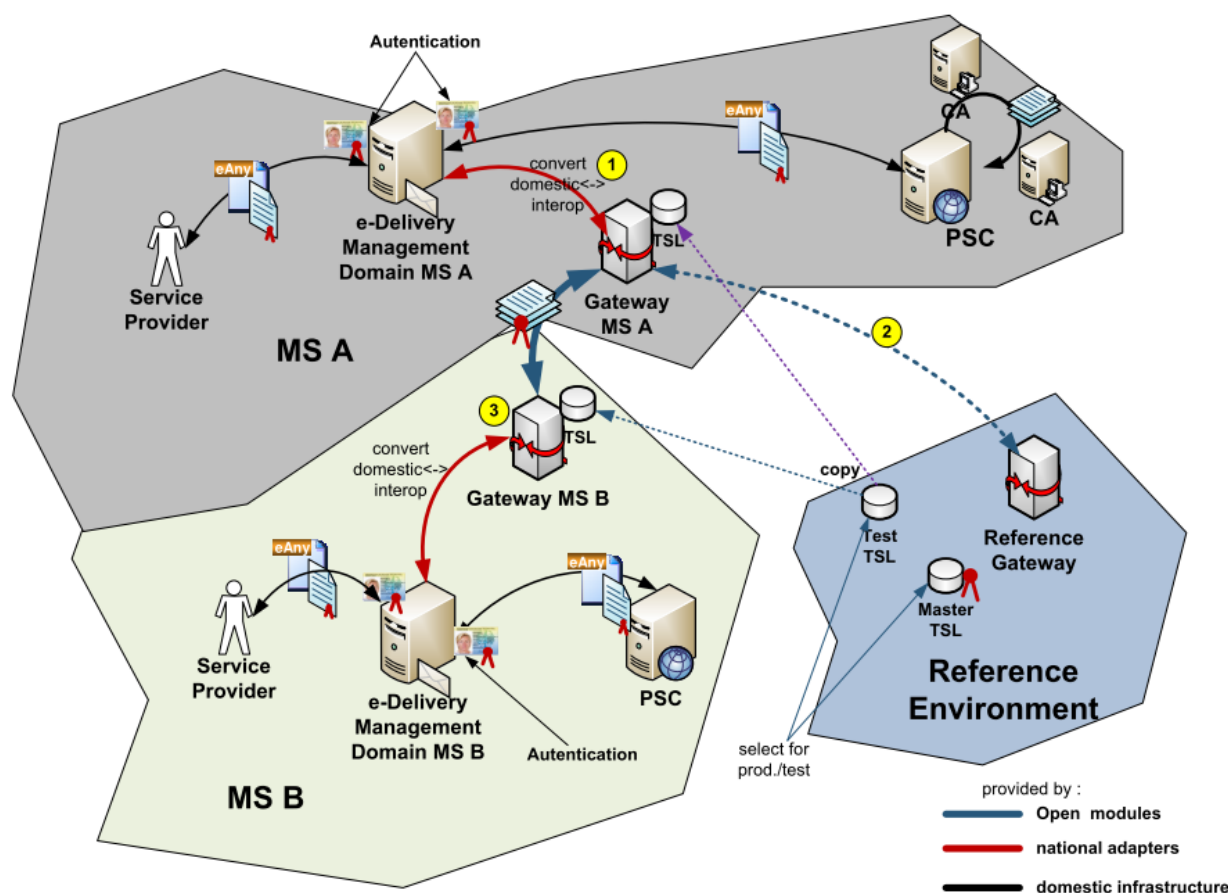


Figure 2: eDelivery components in context of domestic MD's

In the above figure, communication lines marked

- black represent the assumed infrastructure in place in the MS
- blue: those covered by Open Modules provided with this deliverable
- red: this national/domestic adapters must be set up.

Obviously, the integration process has to start with the specification and implementation of the adapter of a specific solution to be connected to the SPOCS eDelivery network (bullet 1 in above figure). The generic Gateway modules provide basis plausibility and scheme conformance checks for message conversion initiated by the domestic adapter's implementation (see section [4.2] and the Open Modules Javadoc). The test version of the SPOCS TSL should already be used in this phase¹³.

Step 2, marked by bullet 2 in above figure, is the recommended one to "simulate" message exchange with a foreign eDelivery infrastructure without directly having to involve such. Using the tests provided here, will help very much to harden specific adapter implementations, before advancing to step 3 below. The Gateway deployed in the Reference Environment is configured to serve as a simulation one, accepting and emitting messages of type REMDispatch as well as

¹³ Deployed in the reference environment at: http://wp3.spocs.bos-asp.eu/TSL/SPOCS_TEST_recent.xml (see SPOCS Wiki for update of contents and location)



REMMDMessage (Evidences). Again, the test version of SPOCS TSL should be used. For details, see section [4.2] and according hints for developers in the SPOCS Wiki:

<https://dev.spocs.bos-asp.eu/wiki/index.php5/WP3Details#eDelivery>.

Finally, intensive test message exchange with foreign eDelivery solutions should be done before entering the production phase (step marked by bullet 3). It's a matter of mutual alignment of involved test partners to define according test details. WP3 members managing the Reference Environment including the TSL test version will help (e.g. to set up according entries). SPOCS Jira must be used to signal according activities and support needed.

3.3. eSafe Specialities

The eSafe Open Modules are provided with the focus to support the integration of eSafes for attaching documents to an application on the PSC, according the specification delivered before.

3.3.1. Contents of the Development

The development of the modules includes components for the PSC and components for the eSafe. Therefore, the parts of the Open Modules of the WP3 deliverable D3.3 are called *eSafe Open Modules* and *PSC Open Modules*. These modules are basically Java libraries that have to be integrated with the *PSC portal application* and the *eSafe portal application*. The portal applications gain benefit of the modules' functionality by calling the respective APIs.

The Open Modules do have a dependency to the TSL modules developed in WP3 as well as the OCD container implementation developed in WP2. The eSafe Open Modules project files include the current TSL library and the current OCD library already fully integrated. However, they may be subject to changes as these libraries have a lifecycle of their own.

For testing and demonstration purposes of SPOCS scenarios the Open Modules are provided together with a Demo PSC and a Demo eSafe package. However, those components are not needed to be integrated into the national infrastructure.

During development, minor changes were made for parts of the deliverable D3.2 with respect to the eSafe specification. Therefore some revised documents from the previous deliverable are added to the current deliverable. Specifically some chapters of the main document as well as the appendix 4 with operations in detail and the eSafe WSDL files were updated. The OCD example is not included anymore since it is part of the WP2 deliverable D2.3.

3.3.2. Recommended Integration Process of the Modules

The following figure provides a brief overview of the scenario which is necessary for testing the eSafe modules. As mentioned earlier most of the parts are supported from the Reference Environment which can be used for the first integration tests with any national implementation.

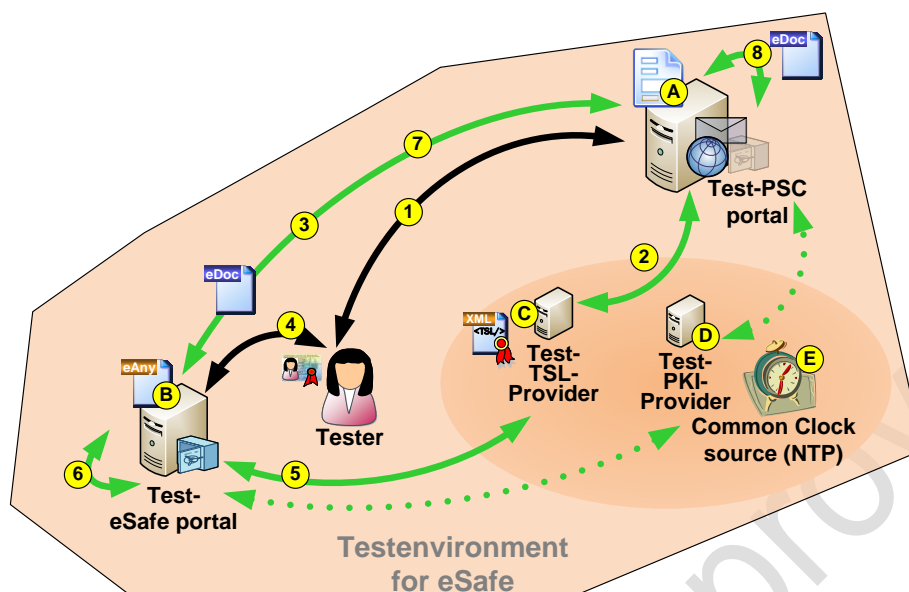


Figure 3: Overview of SPOCS scenario within a Test-environment

Marks 1 to 8 are indicators for requirements according to the SPOCS process for attaching documents with an eSafe (in chronological order), Marks A to E are indicators for requirements where components need to provide functionalities in order to support the SPOCS process.

PSC test portal

- Needs an SSL/TLS certificate (mark A) and a TSL entry (mark C)
- Integration of PSC Open Modules from WP3 (mark A) for the exchange of documents with the eSafe (mark 3)
- Provides a simple form based application for a SP (mark 1) and a simple verification of the attachment retrieved from the eSafe (step 8)

eSafe test portal

- Needs an SSL/TLS certificate (mark B) and a TSL entry (mark C)
- Integration of eSafe Open Modules from WP3 (mark B) for the exchange of documents with the PSC (mark 3)
- Provides a simple way to provide documents in a personal eSafe with various document sizes (mark B) and a simple UI for the selection of documents (marks 4 and B)

TSL-provider

- Provides an online test TSL according the SPOCS specification (marks 2 and 5)
- Provides administration mechanisms in order to change TSL entries and/or different available TSLs (mark C)



Common Clock Source

- Provides the common time which portals can use via NTP¹⁴, can be any external source as well (mark E)

PKI-provider(s)

- Provides issuing, verification and revocation mechanisms for certificates used according the SPOCS specification for SSL/TLS and TSL certificates (mark D)
- Usage of external providers possible: Necessity depends on the certificates used in the Reference Environment

3.3.3. Integration of the Modules and Implementation of the SPOCS Process

Integration process

The following major steps are proposed in order to support an efficient and successful integration process (a more detailed technical overview is provided in section 4.3.4):

1. Each piloting country and partner should get the modules implementation needed, depending on the component (PSC or eSafe).
2. Configuration of the modules
3. Integration and building of the modules into the national implementation
4. Applying the new service for the TSL with the proper TSL details (update of the TSL in the Reference Environment needed)
5. Integration tests with the Reference Environment. Therefore the partner component (for an eSafe it is the PSC and vice versa) can be used from the Reference Environment in combination with the Demo PSC/eSafe (the components Test-eSafe portal/Test-PSC portal in the figure above).
6. Further integration tests as planned.

The next part describes the major tasks a piloting country or any other partner has to consider when integrating a PSC and/or eSafe solution. The prerequisites and requirements are also provided in D3.2 (see chapter 4 in D3.2).

Must-Knows for PSCs when implementing the SPOCS-process

- The PSC has to provide the SPOCS process (see Appendix E of Deliverable D3.2) and offer the possibility to attach documents by integrating the Service Provider's personal eSafe; currently the PUSH model is supported only.
- The PSC has to list the offered services in the TSL according to the specification
- The PSC has to offer the selection of eSafes on the basis of the eSafes listed in the TSL
- The PSC has to handle the application together the received OCD-container, which contains the documents to be attached

¹⁴ For details see <http://www.ntp.org/>



- The PSC has to use the provided PSC Open Modules or implement the services of the document transfer specification

Must-Knows for eSafes when implementing the SPOCS-process

- The eSafe has to offer an entry point, where the Service Provider is directed to and does the authentication first, before selecting the documents
- The eSafe has to offer a selection possibility of documents stored in the Service Provider's personal eSafe
- The eSafe has to use the provided eSafe Open Modules or implement the services of the document transfer specification

Pending EC Approval



4. Technical Overview on Open Modules

4.1. SPOCS TSL

TSLInterface is a Java library that offers a simple interface towards a collection of TSL documents.

4.1.1. Basic Elements

These documents cover a short description of the basic interface elements. Please refer to Javadoc for details.

Class `eu.spocseu.tsl.TSL`

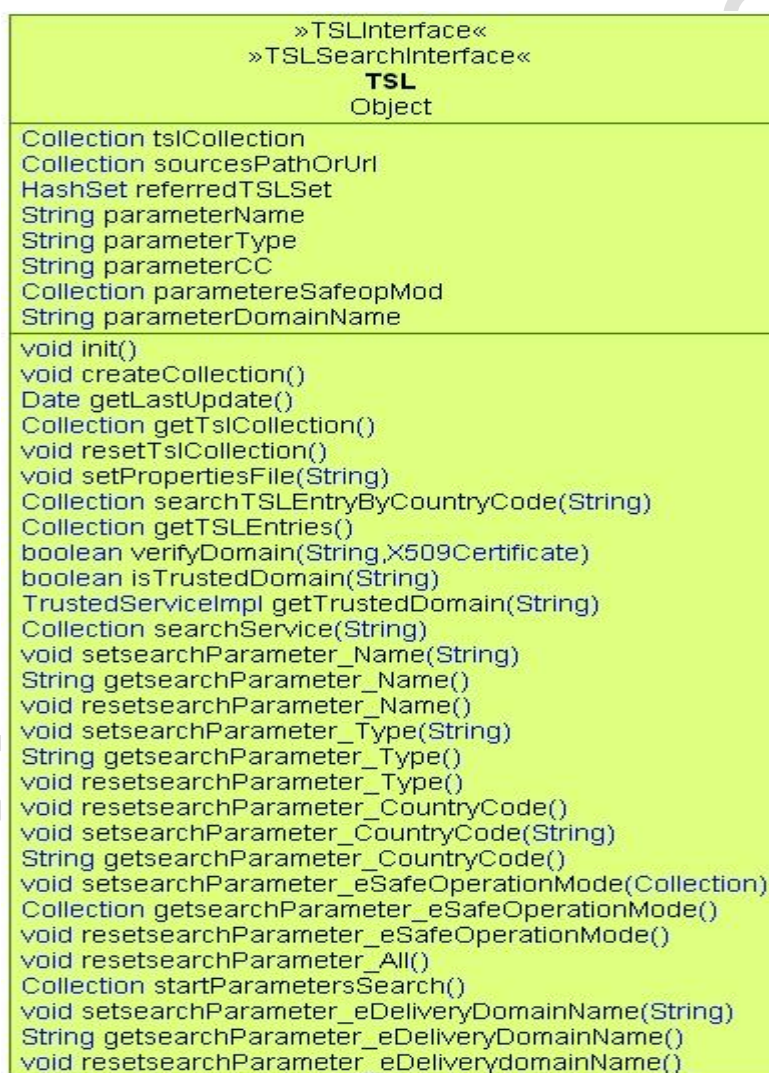


Figure 4: Class TSL

The base class to use, containing the entire TSLCollection, which holds the loaded TSLEntries and offer the methods to initialize the collection.



The constructor needs the correct properties source to initialize the interface.

They can be supplied in two ways:

- through a local file in xml format
- by a properties variable parameter.

The load process is started using the static `init()` method.

To refresh the stored `TSLCollection` directly at runtime, it is also possible to supply the properties needed, i.e. the list of the URI of the TSLs to be loaded, and then calling again the `createCollection` method.

The TSL object offers methods to get the following objects:

- the current stored `TSLEntry` objects collection
- the last update timestamp of the collection

The TSL object implements a `SearchInterface` to search for services. In addition to this, the TSL object offers methods to search the entire `TSLEntry` collection for:

- get the service that manage a specific `TrustedDomain`, supplying the `TrustedDomain` name to search as a string
- verify a `TrustedDomain` against a supplied `DigitalId`, supplying the `X509Certificate` and the `TrustedDomain` name to search as a string
- extract a specific `TSLEntry` by its country code

**Class** `eu.spocseu.tsl.TSLEntry`**Figure 5: Class TSLEntry**

Represents a single TSL entry of the collection loaded.

Contains: a list of TSP in form of `eu.spocseu.tsl.TrustedProviderImpl` objects.

For each TSLEntry the other properties available are:

- the TSL country code
- the TSLId
- the TSL Scheme Operator X509certificate of the signature node (if any)
- the List of the TrustedProviderImpl objects belonging to this TSL

The **TSLEntry** object implements a **SearchInterface** to search for services (see below).

**Class** `eu.spocseu.tsl.TrustedProviderImpl`**Figure 6: Class** `TrustedProviderImpl`

Represent a single TSP of a `TSLEntry`.

For each `TrustedServiceImpl` the object offers methods to get:

- the `TrustedProviderImpl` name
- the `TSPInformationURI` node content
- the list of `TrustedServiceImpl` objects contained

The `TrustedProviderImpl` object implements a `SearchInterface` to search for services.

SearchInterface(`TSL`, `TSLEntry`, `TrustedProviderImpl`)

NOTE: these three classes all implement the `SearchInterface`.

The interface provides methods to services by means of the following parameters:

- Name (string)
- ServiceType (string)
- CountryCode (string)



- Service eSafeOperationModes (List<string>)
- Service eDeliveryDomainName (string)

The parameter can be set, get and reset with the method present (setSearchParameter_XX, getSearchParameter_XX, reset...)

All the parameter inserted for a specific for a search will be in AND. The content of eSafeOperationModes list will be in OR with the TSL content.

Once set the parameters the searching has then to be started calling the method **startParametersSearch()**.

Class eu.spocseu.tsl.TrustedServiceImpl

TrustedServiceImpl TSPService
boolean isSignatureCertificate() boolean isSSLCertificate() void setIdKeyUsageBit(int) void setCountryCode(String) void setTsp(TrustedProviderImpl) TrustedProviderImpl getTSP() String getServiceName() Collection getServiceSupplyPoints() X509Certificate getDigitalId() String getServiceType() X509Certificate getServiceSignatureCertificate() X509Certificate getServiceSSLCertificate() Collection getSupportedEvidence() Collection getRequestedEvidence() Collection geteSafeOpModes() Collection getSupportedEaddressScheme() String getRealmName() Collection getAuthenticationLevel() String getTSLCountryCode() String getExtensionCountryCode() ArrayList getManagedDomain()

Figure 7: Class TrustedServiceImpl

Represents a single service of a TSP.

For each **TrustedServiceImpl** the object offers methods to get:

- the service name
- the serviceType
- the TSP that owns this service
- the service X509Certificate(s)
 - for a service defined with multiple DigitalId there is also a method to find out the certificate type
- the supported evidence list
- the supported eaddress scheme



- the realm name
- the country code of the TSL to which this service belong.

4.1.2. TSLInterface usage

The set of properties conducting the initialization process.

Properties format

Below an example of the **TSLInterface properties xml** file. Some fields are already explained by the related comment.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <comment>Properties and path to files</comment>
    <!--TSL source section: local path or http url (https not
supported yet)-->
    <entry key="TSLSource">/var/opt/TSL_CZ.xml</entry>
    <entry key="TSLSource1">/var/opt/TSL_BE.xml</entry>
    <entry key="TSLSource2">http://localhost:8080/TSL_IT.xml</entry>
    <!--Trust repository section -->
    <entry key="CertificateRepository">/var/opt/myTrustedRep</entry>
    <!--Verification options section-->
    <!--SchemeSignatureVerification - 0: no verification; 1: verify
signature validity-->
    <entry key="SchemeSignatureVerification">1</entry>
    <!--Scheme certificate verification type- 0: no verification; 1:
validity, integrity and trusted repository check; 2: 1 + online
revocation status-->
    <entry key="SchemeCertificateVerificationType">1</entry>
    <!--Service certificate verification type- 0: no verification; 1:
integrity check; 2: 1 + online revocation status -->
    <entry key="ServiceCertificateVerificationType">1</entry>
    <!--Program options-->
    <entry key="FollowTSLPointer">>false</entry>
    <entry key="ForceCacheUpdate">>false</entry>
    <entry key="TCPConnectionTimeout">5</entry>
    <entry key="UseCRLCache">>false</entry>
    <entry key="UseProxy">>false</entry>
    <entry key="ProxyHost">>false</entry>
    <entry key="ProxyPort">>false</entry>
    <entry key="ProxyUser">>false</entry>
    <entry key="ProxyPassword">>false</entry>
</properties>
```

Properties fields explanation:



TSLSource: here can be inserted the file system path (UNIX /path/to/file/ or Windows x:\Path) or http URL from which retrieve the TSL document. Any number of entry can be inserted in the form “TSLSourcexx” where xx is an unambiguous integer.

CertificateRepository: path to folder containing the trusted root certificates file in DER-encoding (with file extension der, cer or crt)

SchemeSignatureVerification [values 0/1]: verify TSL signature true/false (not yet...).

SchemeCertificateVerificationType [values 0/1/2]: verify scheme certificate flag¹⁵

ServiceCertificateVerificationType [values 0/1/2]: verify service certificate flag¹⁵

Here:

- 0: no verification
- 1: integrity check plus, for schemeCertificateVerificationType, the trusted root repository check
- 2: bullet 1 plus revocation check

In this first version, an error in the scheme certificate verification process will be logged on the stderr but the TSL will not be discarded.

FollowTSLPointer: read otherTslPointer node and after initialization of local sources download also the referred TSL. No duplicate will be inserted.

ForceCacheUpdate: if init is called externally, by batch script, this flag control if the TSL in the collection should be updated even if already present as TSLId and CountryCode

TCPConnectionTimeout: TSL download timeout (configurable)

UseCRLCache: cache downloaded CRL locally

UseProxy: http proxy settings (configurable; according operational environment)

¹⁵ The two entries refer to two different types of certificates contained in the TSL, the one with which the TSL is signed (SchemeCertificate) and the Services certificate, that can be automatically checked while reading in the TSL's content. This second kind of check was listed among the requirements.



A. For closer orientation, a code usage example is given in

Pending EC Approval



Appendix D – TSL Interface Code Usage

Pending EC Approval



4.2. Generic eDelivery Gateway

The eDelivery generic Gateway offers a simple interface to connect domestic eDelivery solutions to the SPOCS interconnect infrastructure. According to the specifications developed with D3.2 [1], the generic eDelivery Gateway offers

- Functionalities to convert messages from/to domestic message formats to the one defined by the SPOCS Interconnect protocol
- Routing and mutual authentication between Gateways, based on the Web Services stack implementation Metro¹⁶ and the SPOCS TSL as described in the specification.
- Generation and validation of Evidences according the profiling made in D3.2 for ETSI TS 102 640 [4]; conversion of Evidences to domestic format for delivery receipt functionality
- Generation and validation of SAML token, used on the interoperability layer to attest sender and recipient authenticity; API to convert these SAML token to authentication tokens/information used in respective domestic solutions.

4.2.1. Packages Overview

The whole source code of the eDelivery project can be found in the SPOCS Subversion folder eDelivery-Gateway.

The main project eDelivery contains the sender side Gateway client API and the services needed receiving Gateway side. All classes belong to the package `eu.spocseu.edeliverygw` and its subpackages.

Structure of the main project, to be found in the SVN directory

EDelivery-Gateway\EDelivery-API.

eDelivery Open Modules package name	Description
<code>eu.spocseu.edeliverygw.callbackhandler</code>	Implementations of Metro interfaces
<code>eu.spocseu.edeliverygw.configuration</code>	Configuration classes for the API and the recipient side
<code>eu.spocseu.edeliverygw.evidences</code>	Façade for evidences, represents the REM Evidence schema view
<code>eu.spocseu.edeliverygw.messageparts</code>	Partial messages objects
<code>eu.spocseu.edeliverygw.messages</code>	Façade for messages to be generated by Gateway in role "sending GW"
<code>eu.spocseu.edeliverygw.process</code>	Processing classes for Gateway in role "receiving GW"

¹⁶ <http://metro.java.net/>



eDelivery Open Modules package name	Description
<code>eu.spocseu.edeliverygw.service</code>	The Web Service classes
<code>eu.spocseu.edeliverygw.transport</code>	Classes for the transport or https layer

Table 1: eDelivery Open Modules packages

Detailed information for each class is given in the API Documentation, available online at:

[https://dev.spocs.bos-asp.eu/hudson/job/eDelivery/eu.spocseu\\$EDelivery-API/](https://dev.spocs.bos-asp.eu/hudson/job/eDelivery/eu.spocseu$EDelivery-API/)

4.2.2. Package messages

The package `eu.spocseu.edeliverygw.messages` contains the most important classes to build up and dispatch messages (Gateway in sender role).

Four classes are available for generating and sending the according message types REMDispatch and REMMDMessage (Evidence variants as defined by the specification):

1. `DispatchMessage`
2. `AcceptanceRejectionByRecipientMessage`
3. `DeliveryNonDeliveryToRecipientMessage`
4. `RetrievalNonRetrievalByRecipientMessage`

In the constructor of each class most necessary parameters must be given. After the creation of the object most values can be changed or added; by means of JAXB access is possible to every single element and attribute as defined for the SPOCS Interconnect message schema. For setting the most common parameters, these classes offers some convenience methods. More of such convenience methods will be made available according to needs raising in the phase of implementation of national/domestic adapters.

After complete generation of the message objects, the message can be send to the target Gateway with the `send*` methods of each message object. After calling the `send*` method the eDelivery API will look into the TSL entry to find the address of the according recipient Gateway. If this Gateway instance can be located, the REMDispatch or REMMDMessage is sent in the body of a SOAP message, using the Metro Web Service framework.

4.2.3. Package messageparts

The classes in package `eu.spocseu.edeliverygw.messageparts` must be used to build up specific components of the messages objects described above. The most important classes (according to the sub-elements of Interconnect Protocol schema) are:

1. `DeliveryConstraints`
2. `Destinations`
3. `EvidenceHolder`
4. `MsgIdentification`
5. `Normalized`
6. `Original`



7. Originators

An example of using the classes provided in the packages `messages` and `messageparts` is given in section 4.2.5 below.

4.2.4. Further Objects

In the following, we only mention the most important classes. Like for these, details for all other classes provided are available in the Javadoc.

4.2.4.1. Configuration class

This class is used to configure an eDelivery Gateway instance. All static parameters of national implementation must be configured in a property file, which is parsed by the Configuration class.

4.2.4.2. Interface SpocsGWInterface

`SpocsGWInterface` must be implemented for message acceptance and conversion functionality of a domestic eDelivery Gateway.

This interface provides following three methods.

1. Processing of incoming REMDispatch:

```
public EvidenceHolder processDispatch(DispatchMessage message)
    throws NonDeliveryException, RelayREMDRejectionException;
```

2. Processing of incoming REMMDMessage (Evidence):

```
public void processEvidence(EvidenceHolder evidence)
    throws EvidenceException;
```

3. Will be called at startup time:

```
public void initOnStartup();
```

In the case of incoming messages the Metro framework handles security related functionality of the SOAP message, before control is given to the generic part of the Gateway. After checking signature elements and message structure as defined by the SPOCS Interconnect protocol, the generic Gateway calls the above shown methods of the national implementation for further specific national message handling.

4.2.5. Hints to implement Adapters to domestic eDelivery solution

The SVN project `EDelivery-Gateway/MDTemplate/` is provided as comfortable starting point for the implementation of national/domestic adapters,.

The project should be copied to a subproject which is intended for the domestic adapter implementation. For each message type, the MDTemplate project contains a sample implementation with a lot of inline documentation. All client message samples can be found in the package `eu.spocseu.gw.client`. The samples can be used as template or as source code sample.



The following sample shows how to create and send a simple REMDispatch type message:

```
// These values will be set on the from/electronicAddress. Some of these
// values will be put into the SAML token.
Originators ori = new Originators("rl@localhost-spocs-edelivery.eu",
    "Lindemann, Ralf");
// The destination address will be used to look into the TSL and to
// address the message to the next Spocs gateway with the configured
// credentials.
Destinations dest = new Destinations("ow@localhost-spocs-edelivery.eu",
    "Wilko Oley");
SAMLProperties samlProperties = new SAMLProperties();
samlProperties.setCitizenQAAlevel(1);
samlProperties.setGivenName("Ralf");
samlProperties.setSurname("Lindemann");

// Some detailed information about the message
DeliveryConstraints deliveryConstraints = new DeliveryConstraints(
    new Date());
DispatchMessage dispatchMessage = new DispatchMessage(
    deliveryConstraints, ori, dest, new MsgIdentification("initMsgId",
    new String[] {}), samlProperties);
// All the created objects can be configured in more details by getting
// the underlying JAXB objects with the related getXSDObject
// As an example a message timeout value will be changed.
deliveryConstraints.getXSDObject().setObsoleteAfter(
    SpocsFragments.createXMLGregorianCalendar(3));

String message = "Here the msg text now, this could be xml of course";

Normalized normalized = new Normalized();
normalized.setInformational("mySubject",
    "Here the SPOCS content for testing.");
normalized.setText(message, Normalized.TEXT_FOMATS.TEXT);

Original original = new Original(message.getBytes(), "text/plain");
dispatchMessage.setContent(normalized, original);

DispatchMessageResponse response = dispatchMessage
    .sendDispatchMessage();
LogHelper.LogPrettyIncomingMessage(response.getXSDObject());
```

For further information please have a look into the API-Doc and the inline documentation of the samples.

Hints to for implementing the domestic Gateway message reception:

The "MDTemplate" project provides the sample class **SampleImpl**. It contains the sample implementation of the recipient Gateway interface; sample code with inline comments explains how it works. The sample project builds the deployment WAR with the Maven "package"



command. After building the WAR file it can be copied into the deployment folder of the Application Server.

Before starting the Application Server, the configuration must be customized and copied to the **conf/spocs** directory of your Application Server. A sample configuration file can be found in the **MDTemplate/conf** directory in file **spocsConfig.xml**. Properties in this file must be changed according local environment, afterwards and copied to directory the conf/spocs directory of the Application Server, together with the certificates and key stores.

Further hints are available online and periodically amended in the SPOCS Wiki: <https://dev.spocs.bos-asp.eu/wiki/index.php5/DevelopersGuide>.

4.2.6. Testing

The modules have been developed and tested with

Component	JBoss
Application Server	JBoss 5.1.0.GA (JDK6 Version)
Web Service Framework	Metro 2.1.1
Java VM	Sun JDK 1.6.0_20

Table 2: eDelivery Open Modules tested platforms

The tests are separated into two groups: module tests and integration tests.

4.2.6.1. Module Tests (JUnit)

Following test classes are defined¹⁷:

- **TestConfiguration**
- **TestEvidences**
- **TestMsgIdentification**
- **TestOriginators**
- **TestSpocsFragments**
- **TestDispatchMessage**

These module tests don't need any server to be tested. The tests are integrated into the Hudson integration build process, where all module tests run without failures before a new version will be committed to the SVN.

For further information about the test methods and the expected behavior please consider the API-Doc of the related test classes.

The success of the tests can be followed after every build in the Hudson Environment for the eDelivery project. This can be found at the URL

¹⁷ WP3 eDelivery is about to extend these tests, considering possible specific requirements from domestic adapter implementations

<https://dev.spocs.bos-asp.eu/hudson/job/eDelivery>.

Following example diagram gives an overview of the last 50 builds and their test success rate:

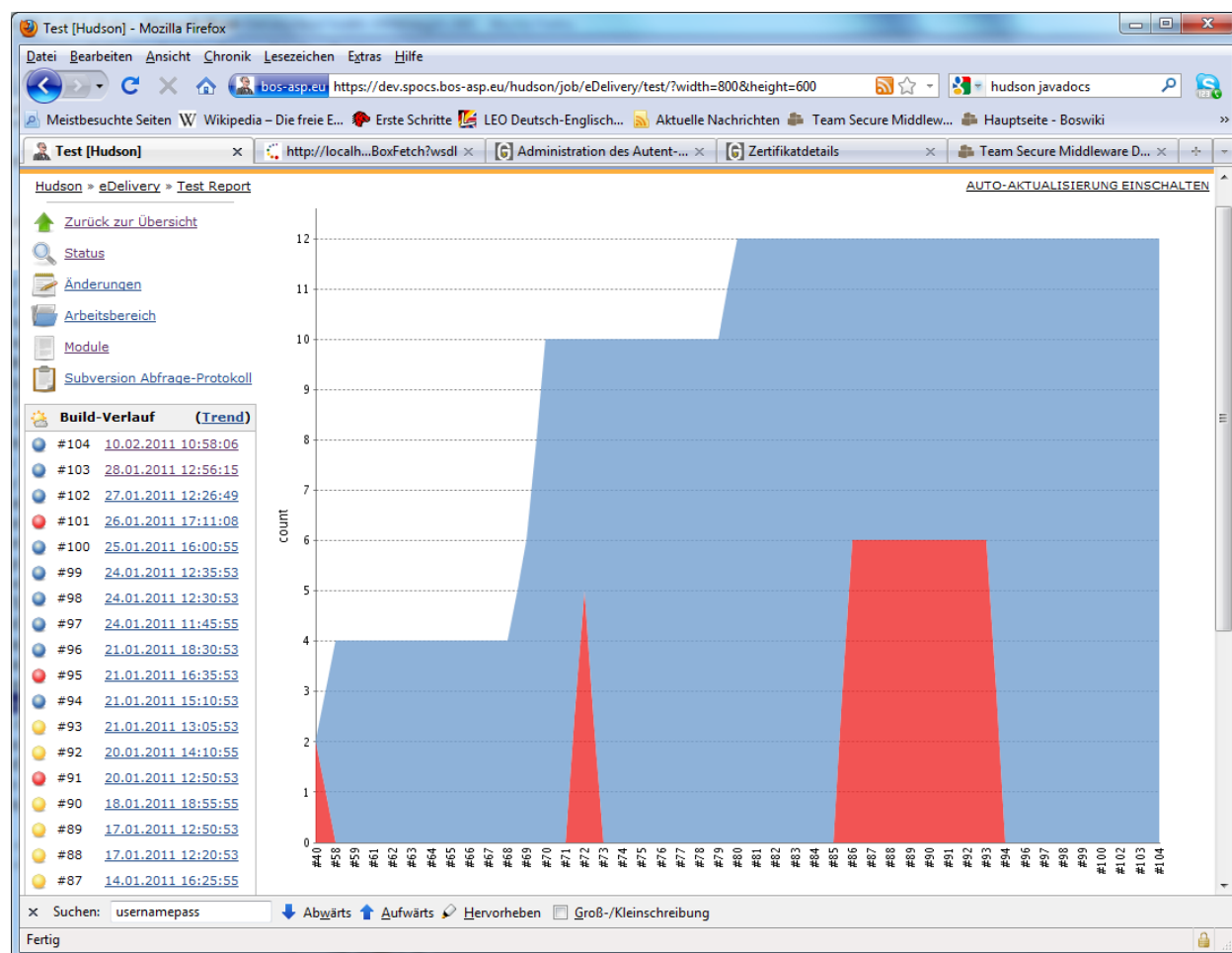


Figure 8: Hudson build process monitoring

4.2.6.2. Support for testing domestic Adapters

4.2.6.2.1. Generic eDelivery Gateway

To test the implementations of the domestic MD Gateway, the Reference Environment provides a generic SPOCS eDelivery Gateway instance which is enabled to provide specific response message (Evidences, error codes) and in addition can be requested to mirror a REMDispatch, thus simulating an incoming initial message for the domestic Gateway instance to be tested.

The requested behaviour can be triggered the requestor. For this purpose, a REMDispatch must be generated and send to the generic Gateway instance. The element **REMDispatch/NormalizedMsg/Informational/Keywords** is used to parameterise the intended behaviour.

A dummy MD must be defined, which is related to generic GW instance in the reference environment, including according TSL entry (routing to this GW instance is handled by domain part of recipients e-address, as defined in specification D3.2).



Details of the **Keyword** element and according triggered function:

@Schema: SpocsTesting

@meaning: SimulateEvidences

With a **Keyword** value of:

- **DeliveryNonDeliveryToRecipient**
Returns a non delivery evidence in the backchannel
- **RelayREMMDRejection**
Returns a non RelayREMMDRejection (Rejection) evidence in the backchannel
- **NonRetrievalByRecipient**
Returns a non RetrievalNonRetrievalByRecipient(NonRetrieval) with a new REMMDMessage some seconds later.
- **RetrievalByRecipient**
Returns a non RetrievalNonRetrievalByRecipient(Retrieval) with a new REMMDMessage some seconds later.
- **AcceptanceRejectionByRecipientMessage**
Returns a non AcceptanceRejectionByRecipientMessage(Acceptance) with a new REMMDMessage some seconds later.
- **AcceptanceRejectionByRecipientMessage**
Returns a non AcceptanceRejectionByRecipientMessage(Rejection) with a new REMMDMessage some seconds later.
- **ReturnDispatch** (to simulate initial REMDispatch entry at domestic Gateway)
Returns a new REMDispatch message some seconds later. Same content, exchanged from/to, sender SAML token etc.

The above list is subject to further amendments, online available in the SPOCS Wiki at:

<https://dev.spocs.bos-asp.eu/wiki/index.php5/WP3Details#eDelivery>.

4.2.6.2.2. **Additional eDelivery Testing Gateways**

These two additional Test Gateways offer a comfortable User Interface to send / receive messages from a national one, as well as access to the different logs of Gateways involved in testing.

Details see: https://dev.spocs.bos-asp.eu/wiki/index.php5/WP3Details#Using_the_UI-based_Test_Gateways_provided_by_Lithuania_2F_JMSYS



4.3. Generic eSafe Connector Modules

4.3.1. eSafe Open Modules Release Notes

The current version of the eSafe Open Modules is known as “version 0.9”. The modules implement all the specified functionality regarding the specified eSafe document exchange protocol with the following exceptions:

- Only the document exchange principle PUSH is supported. The WP3 members and the pilot partners have agreed to implement the PUSH principle with the Open Modules only.
- OCD containers are only simulated by a WP3 internal test container (“OCL Container”, a lightweight OCD like ZIP file). The currently available WP2 software shows integration problems and does not yet support digital signatures and document encryption. While most of the OCD functionality is expected to be ready within the next month the document encryption feature has been postponed for a later SPOCS phase.

The modules have been developed and tested mainly with

Component	Tomcat	JBoss
Application Server	Apache Tomcat 6.0.29	JBoss 5.1.0.GA (JDK6 Version)
Web Service Framework	Metro 2.0.1	JBoss internal framework
Java VM	Sun JDK 1.6.0_1	Sun JDK 1.6.0_1

Table 3: eSafe Open Modules tested platforms

As the software realizes the Web Services with standard JAXB annotations and does not use any proper Application Server APIs, the adaptation to other Application Servers should be fairly simple. Using JDK5 requires at least updated JAXB / XML libraries.

4.3.2. eSafe Open Modules Delivery Items

The software is provided in a subversion repository with the following address:

<https://svnnext.bos-bremen.de/SPOCS/AllWpImplementation/ESafeDocx>

With the subfolders → branches (subfolder for branches, not used yet)
→ tags (tagged versions, e.g. 0.9)
→ trunk (current development)

The trunk folder contains the head revision (the latest versions) of all sources, documentation packages and supplemental tools (not official release items, provided as ‘AS-IS’), while the tags folder references items of a specific release (e.g. 0.9)

Under the trunk subfolder various subprojects can be found.

Block	SVN Repository Folder	Description
Documentation	SPWP3s ¹⁸ _Documentation	<ul style="list-style-type: none"> Enterprise Architect Project describing the design of the eSafe document exchange protocol and the Open Modules implementation "How To" documents JavaDoc Explanation of configuration files
Maven build	SPWP3s_eSafe_Build_Maven_Main	Top level Maven build project
PSC Demo Portal		
PSC Demo Portal EAR Maven projects	SPWP3s_PSC_Demo_Portal_EAR_Build_JBoss5	Maven EAR file project for JBoss5
	SPWP3s_PSC_Demo_Portal_EAR_Build_Tomcat	Maven EAR file project for Tomcat
PSC Demo Portal EAR Eclipse project	SPWP3s_PSC_Demo_Portal_EAR	PSC Demo Portal EAR configuration files
PSC Demo Portal WAR Maven projects	SPWP3s_PSC_Demo_Portal_EAR_Build_JBoss5	Maven WAR file project for JBoss5
	SPWP3s_PSC_Demo_Portal_EAR_Build_Tomcat	Maven WAR file project for Tomcat
PSC Demo Portal WAR Eclipse project	SPWP3s_PSC_Demo_Portal_JEE	PSC Demo Portal sources and configuration files
PSC Module		
PSC Module Eclipse project	SPWP3s_PSC_Module_JEE	PSC Module sources, configuration files and Maven build script
eSafe Demo Portal		
eSafe Demo Portal EAR Maven projects	SPWP3s_eSafe_Demo_Portal_EAR_Build_JBoss5	Maven EAR file project for JBoss5
	SPWP3s_eSafe_Demo_Portal_EAR_Build_Tomcat	Maven EAR file project for Tomcat
eSafe Demo Portal EAR Eclipse project	SPWP3s_eSafe_Demo_Portal_EAR	eSafe Demo Portal EAR configuration files
eSafe Demo Portal WAR Maven projects	SPWP3s_eSafe_Demo_Portal_EAR_Build_JBoss5	Maven WAR file project for JBoss5
	SPWP3s_eSafe_Demo_Portal_EAR_Build_Tomcat	Maven WAR file project for Tomcat
eSafe Demo Portal WAR Eclipse project	SPWP3s_eSafe_Demo_Portal_JEE	eSafe Demo Portal sources and configuration files
eSafe Module		
eSafe Module Eclipse project	SPWP3s_PSC_Module_JEE	PSC Module sources, configuration files and Maven build script
Supplemental tools	SPWP3s_Supplemental	<ul style="list-style-type: none"> Code and documentation templates Certificate generation tools Application Server configuration utilities Sources of non Apache external libraries

Table 4: eSafe Open Modules delivery items

The folder structure may be subject to changes during the next phases, depending on the feedback that is given during the pilot integration phases.

¹⁸ SPWP3s stand for **SPOCS Work Package 3 eSafe Integration**

4.3.3. eSafe Open Modules technical Architecture

Both the PSC and the eSafe open module are provided as Java libraries that are to be integrated in a Java-based portal application. The portal applications have to implement the user interfaces as described in the SPOCS eSafe document exchange protocol specification and call the module's API as described

- in the technical design diagrams that are provided with the modules as a Sparx Enterprise Architect project file (a free viewer is available at the Sparx web site),
- in the modules' public API JavaDocs, also provided together with the modules.

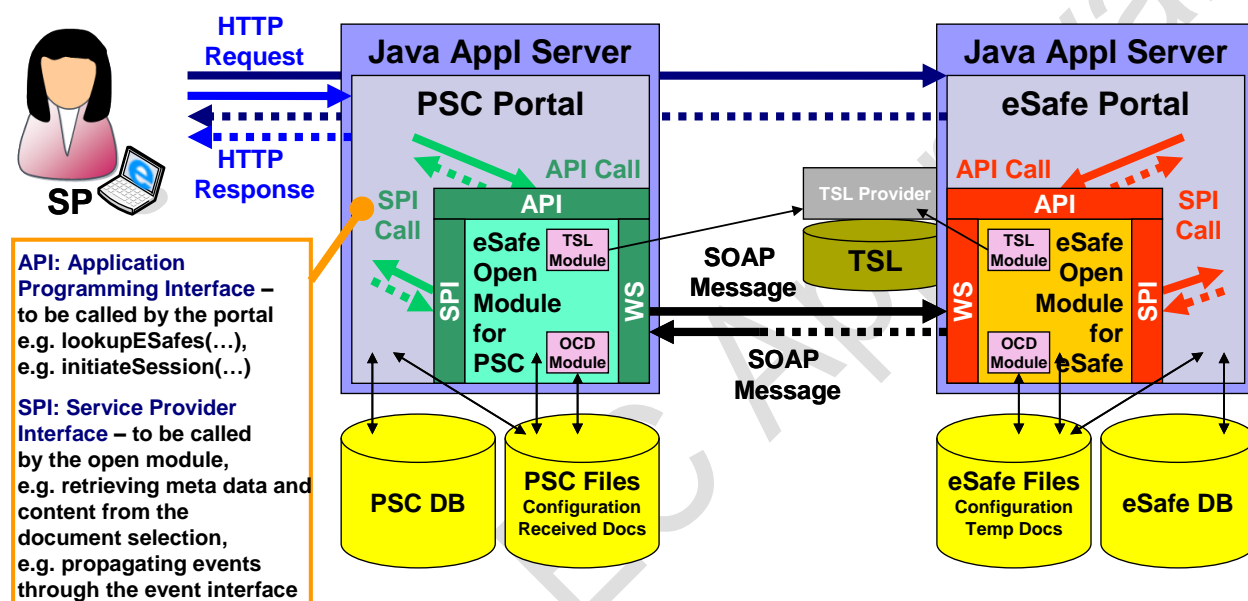


Figure 9: eSafe Open Modules technical architecture

Through the SPI, also described in the technical design diagrams and in the modules' public API JavaDocs, the Open Modules can provide system events or access data provided by the portal (e.g. the selected eSafe documents).

4.3.4. Overview on the Procedure for Integration

The following list gives an overview of the most important steps of integrating the eSafe Open Modules with a real PSC or with an eSafe portal system.

1. Integrate the delivered modules/libraries

- Configure the basic module's settings in the appropriate configuration files (serviceproperties.xml, serviceproperties.keystore, sessionproperties.xml), e.g.
 - Address of the TSL provider
 - portal's name
 - web site URL
 - certificates



- folder for storing document transfer packages
 - maximum document transfer package size
 - transfer options (e.g. frame size)
 - timeouts, etc. (see module configuration files documentation for further details)
- b. Register the portal's UI entry points (URL templates) relevant to the eSafe document exchange protocol in the Open Modules' configuration files
 - c. Include the module in the application startup procedure
 - d. Configure the server environment like SSL and firewall settings

2. Extend and enable SPOCS functionality

- e. Implement the SPOCS-specific UIs
- f. Use the module's API (e.g. session object) for accessing the module's functionality
- g. Implement the module's SPI (e.g. DocumentSelection) for
 - providing the relevant data (selection and provision of documents, metadata, etc.) and for
 - implementing optional hooks (e.g. event listeners) depending on the portal's role (PSC or eSafe)
- h. Publish the open module's web services (e.g. registering in the portal's web.xml)

3. Initiate entry in TSL for the component

- i. Each role (PSC, eSafe) needs to be included in the TSL
- j. Resources required
 - Standard TSL attributes
 - Service name → should be unique, eg. qualified with the domain
 - Service digital identity → Trustworthy SSL Certificate
 - Service Supply point → URL of the InfoService WSDL
 - countryCode
 - document transfer principle (PUSH, PULL)
(note: delivered modules support and provide PUSH principle)

Note: Detailed explanations of the activities and the various configuration files and their settings can be found in the documentation inside the subversion repository.



4.3.5. Testing the eSafe Open Modules

As mentioned above, the eSafe Open Modules have been developed and tested mainly using Apache Tomcat 6.0.29 and JBoss 5.1.0 GA both using JDK 6.

4.3.5.1. JUnit Tests

For automatic component and simple integration testing the sources currently include 70 JUnit test classes organised into various categories by

- Test suites – organised by functional areas (following java package clusters)
- Test class hierarchies – organised by technical requirements (no configuration needed, running with the standard configuration, running with a special test configuration)

During the next phases that cover bug fixing and improvements by feedback of the SPOCS partners, the number of JUnit test cases will further increase.

The eSafe Open Modules source code repository (subversion) contains two log files of recent JUnit test runs (one for the PSC open module and one for the eSafe open module) in the documentation section. Note that the exceptions that are recorded in the log files do not indicate test failures. They indicate constructed error situations that have been successfully caught.

4.3.5.2. Scenario Test with the Demo Portals

As discussed previously, the Open Modules are provided together with a demo PSC and a demo eSafe package. These demo portals shall help the partners to understand the modules' usage and help testing the own implemented SPOCS functionality using the one of the demo portals as a communication partner simulation.

Once your Eclipse IDE and your Application Server environment have been configured and all the projects have successfully been compiled the demo portals can be deployed and started. Using a local Eclipse development environment you should be able to enter the demo portals by browsing the URLs

PSC: http://localhost.demosystem.eu:8080/SPWP3s_PSC_Demo_Portal_JEE/

eSafe: http://localhost.demosystem.eu:8080/SPWP3s_eSafe_Demo_Portal_JEE/

or an similar URL using the respective HTTP or HTTPS configured port.

Note: the hostname localhost.demosystem.eu is only needed when using the certificates that have been included in the sources. This hostname requires an entry in the local hosts file pointing to your workstation IP address or to 127.0.0.1

The demo portals' look and feel has been inspired by the official SPOCS web site. The PSC Demo Portal home page looks like this.



Figure 10: Demo Portal Home Page

For not getting confused when the user is redirected to the eSafe Demo Portal during the demo application scenario the upper right corner shows the portal identifier.

The section “About Demo” provides some of the configuration information and gives also access to the system’s WSDL files.



SPoCS **PSC Demo Portal**

Home About Demo Demo Application Stress Test Demo

You are here: about

About Demo PSC

Configuration


Identifier

Service Type:	http://uri.spocs-eu.eu/Svctype/PSC/v1
Name:	www.demopsc.eu
Country code:	DE
Site:	https://localhost.demosystem.eu:8081/SPWP3s_PSC_Demo_Portal_JEE/
WSDL:	https://localhost.demosystem.eu:8081/SPWP3s_PSC_Demo_Portal_JEE/index-wsdl.html

Capabilities

Protocol version:	1.0
Document exchange principles:	[PUSH]
Access token transfer strategies (PULL only):	[]
Supported encryption algorithms:	[]
Supported digest algorithms:	[]
Supported signature algorithms:	[]
Transfer frame size:	1000000

Disclaimer

  SPOCS is an EU co-funded project
CIP-ICT PSP-2008-2 n°238935

Copyright 2010 SPOCS. All rights reserved.
Gov2DemOSS is Free Software released under the GNU/GPL License. 

Fertig FoxyProxy: Default

Figure 11: Demo Portal About Demo page



The section “Demo Application” allows running a simple scenario including browsing eSafes of a given country, redirection to a specific eSafe, document selection and transfer of a set prepared documents and randomly created files. Here you see the user has already been redirected to the eSafe and the selected documents are transferred to the PSC.

The screenshot shows the SPOCS ESafe Demo Portal in a Mozilla Firefox browser. The page title is "ESafe Demo Portal". The navigation bar includes "Home", "About Demo", and "Demo Application". Below the navigation bar, it says "You are here: demo".

The main content area is divided into two sections:

- Process steps:** A vertical list of buttons: "START APPLICATION", "ESAFE LOOKUP", "ESAFE SELECT", "ESAFE SESSION INITIATION", "DOCUMENT SELECTION", "DOCUMENT PACKAGING", and "DOCUMENT PUSH TRANSFER".
- Document transfer status:** A yellow box containing the text "TRANSFER_IN_PROGRESS", a progress bar at 75%, and the text "3 of 4 frames transferred" and "3000000 of 3816867 bytes transferred". There is an "Abort Transfer" button.

Below these sections is a table showing log messages:

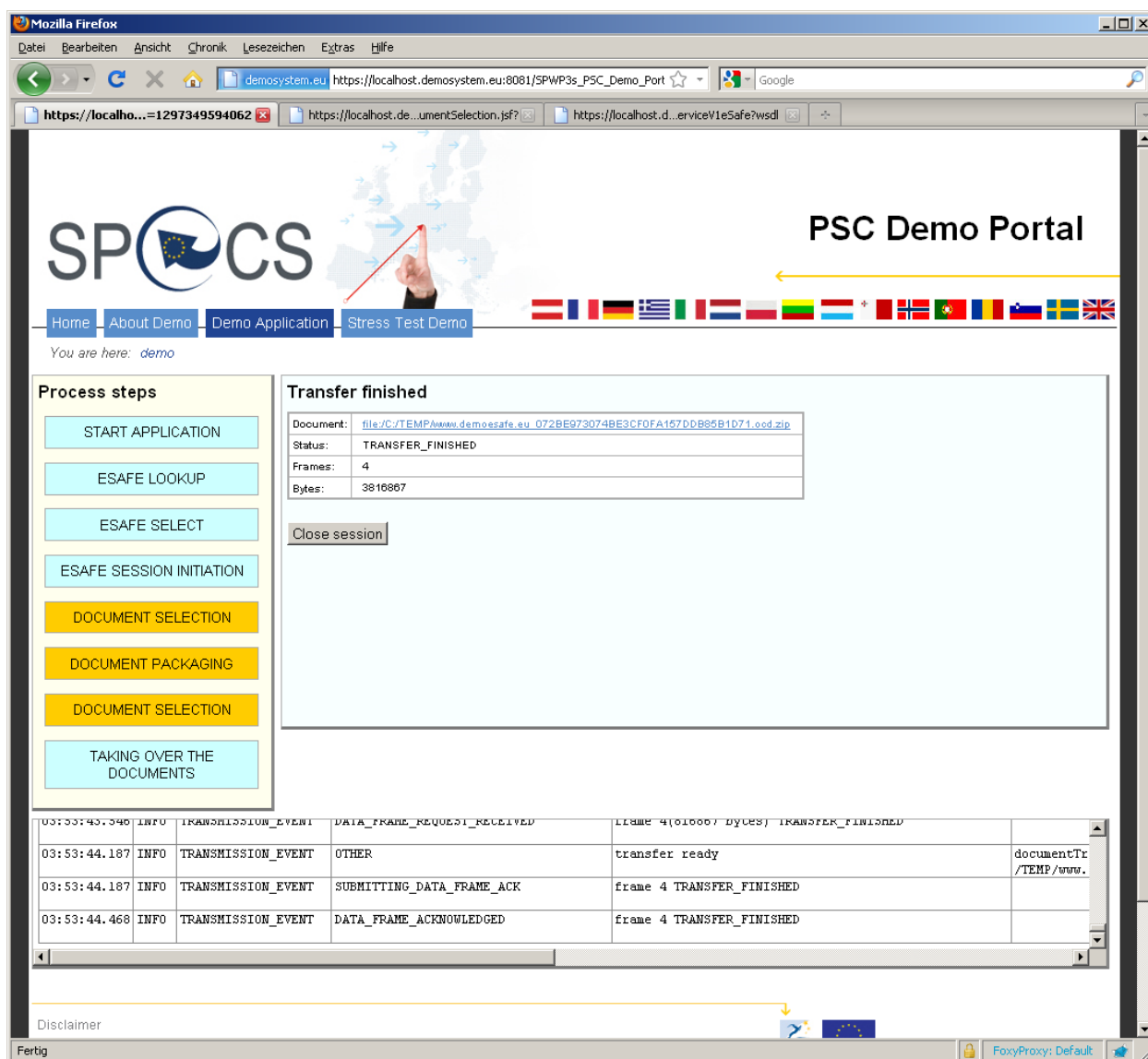
Time	Level	Event	Message	Details
03:53:14.62	INFO	CONNECTION_EVENT	SESSION_INITIATION_ACCEPTED	www.demopsc.eu(DE) ACCEPTED sessionId=1297349594062
03:53:41.593	INFO	DOCUMENT_ACCESS_EVENT	DOCUMENT_TRANSFER_PACKAGE_PREPARED	documentTransferPackage=file:/C:/TEMP/dtp_1297349594062_7171940576546784088.oc1 status=PACKAGE_CRE
03:53:41.750	INFO	TRANSMISSION_EVENT	SUBMITTING_DATA_FRAME	frame 1 TRANSFER_IN_PROGRESS

At the bottom, there is a "Disclaimer" section and a logo for SPOCS, which is an EU co-funded project CIP-ICT PSP-2008-2 n°238935.

Figure 12: Demo Portal Document Transfer Status page

While running the test the Eclipse console dumps all the log messages as configured in the module settings. Additionally the web page shows all the events that are reported by the open module through the SPI.

Finally, the demo ends with a page where you can inspect the transferred documents.



SPoCS PSC Demo Portal

Home About Demo Demo Application Stress Test Demo

You are here: demo

Process steps

- START APPLICATION
- ESAFE LOOKUP
- ESAFE SELECT
- ESAFE SESSION INITIATION
- DOCUMENT SELECTION
- DOCUMENT PACKAGING
- DOCUMENT SELECTION
- TAKING OVER THE DOCUMENTS

Transfer finished

Document: file:///C:/TEMP/www.demosafe.eu_072BE973074BE3CF0FA1570DB85B1D71.cod.zip

Status: TRANSFER_FINISHED

Frames: 4

Bytes: 3816867

Close session

03:53:43.346	INFO	TRANSMISSION_EVENT	DATA_FRAME_REQUEST_RECEIVED	FRAME 4(3816867 BYTES) TRANSFER_FINISHED	
03:53:44.187	INFO	TRANSMISSION_EVENT	OTHER	transfer ready	documentTr
03:53:44.187	INFO	TRANSMISSION_EVENT	SUBMITTING_DATA_FRAME_ACK	frame 4 TRANSFER_FINISHED	/TEMP/www.
03:53:44.468	INFO	TRANSMISSION_EVENT	DATA_FRAME_ACKNOWLEDGED	frame 4 TRANSFER_FINISHED	

Disclaimer

Fertig

Figure 13: Demo Portal Transfer Finished page



4.3.5.3. Stress Test with the Demo Portals

The section “Stress Test Demo” allows running a multithreaded scenario with a random number of threads. In this case the Demo eSafe always returns a fix set of documents.

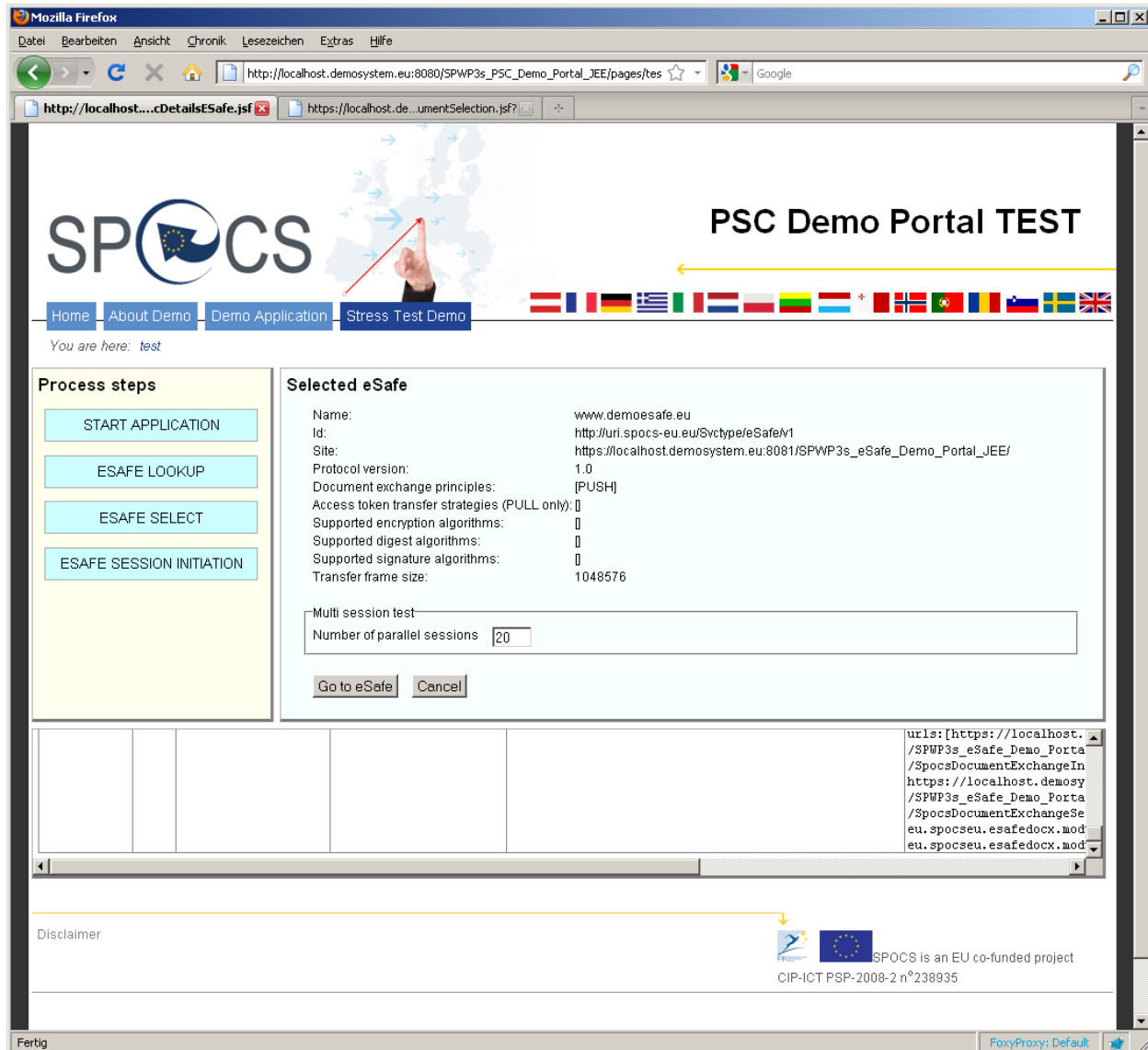


Figure 14: Demo Stress Test Demo page

While running the test the Eclipse console dumps all the log messages as configured in the module settings. Additionally, the web page shows all the events that are reported by the open module through the SPI. Note that the created load and the continuous refresh of the test status page may lead to log messages reporting that some Java Server Faces Events cannot be executed in time. This is due to the current simplicity of the stress test implementation.



5. Conclusions

The current deliverable D3.3 is based on the results of the deliverables D3.1 and D3.2 and provide the audience with an overview of the Open Modules, developed according to the specifications. Since WP3 covers two major components, eDelivery and eSafe, there are separate implementations available.

As specified in the dedicated specification documents, the current implementations are highly focussed on interoperability aspects and provided via the European Union Public Licence (EURL). This focus was targeted in the specification itself and further extrapolated within the chosen development platform and Reference Environment as well. The current deliverable provides an overview of the modules themselves and what public available tools and frameworks were used. For quality purposes, the modules were also tested during the development. Further integration tests need to be processed when integrating the modules within the national solutions. The deliverable also covers the usage of a SPOCS-TSL (Trust-service Status List), which builds the basis for the chosen trust model. The various specialities of the components are provided in section 3, Open Modules Development, Operational Platforms and Testing. eDelivery provides a Gateway which enables the communication between different national eDelivery solutions. eSafe provides modules for the integration of eSafes, in order to exchange documents with PSCs and two Demo-Portals (PSC and eSafe), in order to demonstrate and visualise the SPOCS scenario and communication.

With the technical overview in chapter 4 (Technical Overview on Open Modules), which describes the structuring and summarises the documentation provided, the partners and piloting countries are able to get an overview about the (implementation) results of WP3.

In addition to the development of the Open Modules, a Security Model was developed in order to illustrate security-related concepts by means of a security architecture development process. Since security does not end with the development of the modules itself, the results also cover risks which have to be considered by the Member States' implementations within the integration and operating phase. Section 2, Security Model - Results of Security Architecture Development Process and Formal Modeling & Verification covers the details.

Last but not least, the next phase will focus on the support of the piloting countries. It is the major goal to successfully deploy the modules developed in the SPOCS piloting countries, with the ability to solve the addressed interoperability issues.

And as mentioned in D3.2 already, the proof of the pudding is in the eating. Technically speaking, the modules will be successfully adopted and integrated in various implementations!



References

- [1] SPOCS Project D3.2 Specifications for interoperable access to eDelivery and eSafe systems, http://www.eu-spocs.eu/index.php?option=com_processes&task=streamFile&id=18&fid=699 (last visited on 11th February 2011)
- [2] SPOCS Project D3.1 "Assessment of eDelivery systems and specifications required for interoperability", http://www.eu-spocs.eu/index.php?option=com_processes&task=streamFile&id=18&fid=934
- [3] ETSI TS 102 231, v3.1.2, Electronic Signatures and Infrastructures (ESI); Provision of harmonized Trust-service status information; http://www.etsi.org/deliver/etsi_ts/102200_102299/102231/03.01.02_60/ts_102231v030102p.pdf (last visited on 20th May 2010)
- [4] ETSI TS 102 640-2 V2.2.1, Electronic Signatures and Infrastructures (ESI); Registered Electronic Mail (REM); Part 2: Data Requirements and Formats for Signed Evidences for REM; http://www.etsi.org/deliver/etsi_ts/102600_102699/10264002/02.02.01_60/ts_10264002v020201p.pdf (last visited on 26th March 2012)



B. Appendix A – Acceptance Criteria

	Acceptance criteria	Norm	Process	Priority
1.	<ul style="list-style-type: none"> Conform to SPOCS template 	<ul style="list-style-type: none"> Template issued by Gov2u (12-2010) 	Check against template by WP6	High
2.	<ul style="list-style-type: none"> Language & Spelling 	<ul style="list-style-type: none"> English (UK) 	Review by an English language specialist by PD	Medium
3.	<ul style="list-style-type: none"> Readability <ul style="list-style-type: none"> related target groups <ul style="list-style-type: none"> Executive Summary (e.g. for members of the Commission or reference groups, who expect a rapid overview of the results) Main Part and Conclusion Layout Clearness of the sentences 	<ul style="list-style-type: none"> comprehensible English Clear structure & interesting presentation Highlight of outcomes and advantages Benefits from using WP3 are marked, comprehensible for the technical experts Structured sections especially a hierarchical structure no complicated structure, no nested sentences, - in this context- no unusual vocabulary 	Review by PD Review by each partner and general review by Sub-Leader Team Review by Sub-Leader Team Review by each partner, Sub-Leader Team and PD “	High High High High
4.	<ul style="list-style-type: none"> Consistency with description in DoW 	<ul style="list-style-type: none"> Alignment between this deliverable and the DoW regarding the effected account/statements about objectives and the description of the deliverable D 3.3 	Examine this deliverable and the DoW and compare the effected account/statements. Review by Sub-Leader Team and partners	High
5.	<ul style="list-style-type: none"> Consistency within the document 	<ul style="list-style-type: none"> No contradictions and unnecessary repetitiveness 	To check that there are no contradictions and needless repetitiveness. Review by Sub-Leader Team and partners	High

6.	<ul style="list-style-type: none"> Content is suitable for purpose 	<p>The deliverable fulfils the objectives of the DoW (See list in next paragraph). The deliverable provides the necessary facts to inform the target groups, forms the basis for the preparing and realisation of the piloting phase</p>	<p>To check towards the result list and usability. Reviewed by Work Package Leader and Sub-Leader-Team in collaboration with relevant partners (piloting countries) if needed.</p>	High
7.	<ul style="list-style-type: none"> Contents is fit for use 	<p>Based on this deliverable the piloting phase can be prepared and started</p> <ul style="list-style-type: none"> Overview of WP3 and the integration to SPOCS is described, especially inter WP dependencies are defined Interconnect Protocol components and eSafe Open Modules are integrated with the expected results as described. The documentation recommends information for specialists to handle the integration, The Interconnect protocol and the eSafe open modules implement the specs from D3.2 The interconnect protocol and eSafe Open Modules conform to the conventions of the Reference Environment (coding and documentation). 	<p>To Check towards this result list and usability. Reviewed by Sub-Leader – Team in collaboration with relevant partners if needed (e.g. authors D3.3)</p> <p>The technical inline documentation could be checked by random sample in selected components.</p>	High
8.	<ul style="list-style-type: none"> Commitment within WP 	<ul style="list-style-type: none"> Partners of WP are aware of this acceptance list they had committed before. Commitment by every WP3 partner to the functionality of the Interconnect protocol and deliverable input. 	<p>The experts of the partners are involved in QA-Process. Review by e-mailing and commitment by conference call</p>	High
9	<ul style="list-style-type: none"> Delivered on time 	<ul style="list-style-type: none"> Planning for the Work Package 	<p>Check the Project plan at regular intervals and contact the suppliers to confirm the agreed delivery dates.</p> <p>WP-Leader together with Sub Leader Team</p>	High

Table 5: Acceptance criteria list



Legend:

- Deliverable - a description of the deliverable for which the acceptance criteria must be met.
- Acceptance criterium – a description acceptance criterium
- Norm – a description of the norm that is applied to measure conformance
- Process – a description of the process that is used to test conformance
- Priority – the priority to meet a acceptance criterium (Low = nice to conform to, Medium = important to conform to, High = necessary to conform)

Pending EC Approval



C. Appendix B – Risk List

Threat	Consequence(s)	Measure(s)	Chance	Impact	Risk
1.General risks relating mainly to the development of deliverables					
Provided contributions do not have the sufficient quality and quantity.	Time and resources for revision and re-work are necessary. Running behind schedule. Deliverable will not be submitted on time.	The acceptance criteria list was defined before the start and agreed within the WP. This list contains e.g.: The use of the SPOCS template (deliverable document). Follow the rules and formal structure as determined, agreed and described in the wiki of the Reference Environment. Internal quality checks of experts on important points during delivery (e.g. usability of the result). Early involvement of the Executive Board quality reviews (to save time). Controlling time-line (agreed by the partners in the WP3 meeting) and reminding the partners to meet the deadlines. WP leader monitors the delivery. Co-operation with PD.	M	H	H
Contributions of partners/member states contact persons are not delivered on time			H	H	H
Deliverable is not accepted by PTC, EQM, PD			M	H	H

Threat	Consequence(s)	Measure(s)			
2. Examples for concrete risks relating mainly to the development of deliverables					
<p>Missing internal deadlines because of</p> <p>1. Povision of modules delayed or</p> <p>2. Provision of code contributions is too late.</p>	<p>1. to save time, poor testing and documentation with the consequence of decreasing quality (e.g. in reliability)</p> <p>2. The finalisation of the concerned module is at risk or has reduced functionality or potentially a loss of relevant content or quality aspects.</p> <p>Falling behind schedule.</p> <p>Delay in progress in WP.</p> <p>Partially missing the objectives of the WP.</p>	<p>Utilise the appropriate measures of part 1.</p> <p>Contact the partners and remind them.</p> <p>Monitor the progress frequently.</p> <p>Look for compromises and efficient processing (to save resources).</p> <p>Co-operation with PD.</p> <p>Escalation within project.</p>	M	H	M
<p>New technical problems arise, which have to be solved.</p>	<p>Potentially the module does not fulfil its task correctly or will not be run able or become active.</p> <p>Falling behind schedule.</p> <p>Delay in progress in WP.</p>	<p>Utilise the appropriate measures of part 1.</p>	M	H	M
<p>The test environment (technical testing) does not function.</p>	<p>No testing is carried out.</p> <p>Delay in progress of finalised modules and their connecting together.</p> <p>Missing the deadline.</p> <p>Possible delay of dissemination and pilot implementation.</p>	<p>Utilise the appropriate measures of part 1</p> <p>Check the functionality of RE regularly (follow a pre-prepared control plan).</p>	L	M	L

Threat	Consequence(s)	Measure(s)	Chance	Impact	Risk
Risks relating mainly to the scope of the work package					
The connectors cannot be attached to the 'eDelivey Interoperability Gateway'.	No eTransport is possible. No real interoperability.	Provision of educating material for the pilots and in addition intensive consulting by the partners who developed the open modules (common analyse)	M	H	H
Poor co-operation from technical WP-partners in consulting during the pilot implementation (e.g. lack of resources).	Pilot cannot start on time. Potentially a loss of quality/quantity. Objective missing.	Complete technical manuals for pilot implementation should be produced in parallel with the development and should be used for early education. Further dialogue with organisations. Preparing concise documents for easy response. Escalation, possibly via the PD.	M	H	M
Too few resources of experts to provide the complete functionality.	The module(s) does/do not fulfil the expected task correctly or completely.	Ask other SPOCS partners for support. Establish new schedule to start with a basic version and complete the work step-by-step with reduced availability of developers – but over a longer period. Thus, the pilot can start and the functionality will increase in stages.	M	H	H

Threat	Consequence(s)	Measure(s)	Chance	Impact	Risk
Risks relating mainly to the project in general					
No common agreement about piloting test environment	Pilot functions are not running perfectly, are wrong or are not available. Potential problems in achieving the objectives of the project.	Strong current dialogue between partners, further discussion and decisions based on Athens meetings needed.	M	M	M
The IT infrastructure of the PSC in the piloting countries does not meet to the requirements of the open module, or the agreed process / use case.	Potentially the implementations of cross-over connections are not possible. The SPOCS objectives may not be achieved.	Raising awareness by stakeholders WP5 and WP7 make a concerted effort <ul style="list-style-type: none"> - to clarify the construction of IT-Infrastructure needed and - responsibilities at PSC level 	M	M	M
The responsibilities are not clarified at the PSC level.	Piloting is almost not possible.	<ul style="list-style-type: none"> - and finally the negotiating and trying to achieve an agreement to implement all necessary components 			
Missing project acceptance by European Member States.	Ultimately we would fail to meet the objectives of the project.	Public relations actions as well as clarifying actions of WP6	M	M	M
Too few SPs want to use the pilot solutions.		Ask for feedback from the Member States' view Apply to appropriate marketing actions Integration of new piloting countries (extension), integration of feedback by industry, other member states, other public groups			

Table 6: Risk List



Legend:

- ☐ Threat - Description of a potential danger towards the project.
- ☐ Consequence - Description of the negative effect the threat can have towards the project.
- ☐ Measure - Description of the measures that can be taken to prevent a threat from happening or to reduce negative the effects.
- ☐ Chance - Defines the likelihood of a threat to happen.
- ☐ Impact - Measure of the negative effect on the project.
- ☐ Risk - Chance * Impact, representing the priority.

Pending EC Approval



D. Appendix C – Formal Modelling and Verification Results

This appendix details section 2 “Security Model - Results of Security Architecture Development Process and Formal Modelling & Verification”.

The following related documents have been used:

- SPOCS D3.2
- The SPOCS D3.2 Appendix1: Security Architecture Development Process
- The SPOCS D3.2 Appendix 5: SPOCS TSL Accreditation and Operation Policy
- The SPOCS D3.2 Appendix6: Security Model
- SPOCS D3.3

D.1 Purpose

Our formal analysis of the SPOCS eSafe scenario and Open Module - based on the Security model in “SPOCS_D3 2_Appendix6_Security Model” - has revealed a number of potential security issues/risks.

The following chapter details the assumptions used, as well as the description of possible attacks and proposed countermeasures.

D.2 Exemplary Exploit Scenarios

The exploit scenarios given below are intended as “non-technical” examples on how the revealed weaknesses could be used for fraud or for attacking the availability of the SPOCS interoperability framework.

The detailed description of the attack scenarios is provided in Chapter D.3 “Attack Scenarios and Revealed Risks”.

D.2.1 Impersonation Attack:

- **Attack:**
 - The attacker Mr. X successfully registers a business B impersonating Mr. Smith
 - Mr. X
 - makes expensive purchases using business B or
 - commits fraud with value added tax or
 - commits some other illegal activities Mr Smith is liable for
- **Result:**
 - Mr. Smith as supposed manager of B will be held liable for all of these activities of Mr. X and has to prove his innocence



D.2.2 Denial of Service Scenario:

- **Attack:**
 - Mr. X successfully manages to attack the TSL infrastructure or prevents the distribution of the TSL
- **Result:**
 - All SPOCS components stop working

D.3 Attack Scenarios and Revealed Risks

D.3.1 Assumptions and Consequences

Numerous scenarios have been modelled and analyzed, covering following assumptions:

- No mandatory registration of SPs at PSCs
- No authentication of SPs at PSCs
- Weak registration at SPOCS components
- Weak authentication at SPOCS components

As a result, these scenarios were found to violate the following security objectives

- Confidentiality
- Authenticity / Integrity
- Availability
- Non-repudiation

D.3.2 Risks due to missing or weak Registration and Authentication at PSC

- **Assumption:**
 - There is no common specification for a PSC. In particular, there is no mandatory strong registration and strong authentication requirement.
 - PSC offers access without (or with only weak) registration and authentication
- **Attack: Impersonation Attack – Attacker impersonates a SP**
 - Attacker impersonates a SP and initiates request to honest PSC (possible due to missing registration procedures) or registers and initiates request to honest PSC (possible due to weak registration procedures).
 - Honest PSC offers eSafes to Attacker
 - Attacker chooses his (valid) eSafe account from offered eSafes and chooses a document for upload on PSC



- eSafe offers (signed) document/OCD Container to PSC belonging to impersonated SP
- **Result:**
 - Attacker is able to finalize a correct request with PSC and eSafe, impersonating a SP
- **Reason:**
 - No reliable link between PSC account and real or legal person established
- **Violates:**
 - Authenticity
 - Integrity
 - Non-repudiation
- **Possible Risk Mitigation:**
 - Definition of “non-IT” measures, e.g. business processes and trained personnel, to detect impersonation attacks at the business content level
 - Empowerment of a mandatory PSC specification/policy that requires mandatory registration and a minimal authentication level - or even better, empowerment of a mandatory PSC specification that requires mandatory strong registration and strong authentication.

D.3.3 Risks due to difficulties during the Authentication of SPOCS Components against SP

- **Assumption:**
 - There is no convenient and easy to use method for the SP to verify the authenticity of SPOCS components.
 - In the usage of the Web, sloppy behaviour has established.
- **Attack: Impersonation Attack - Attacker impersonates a PSC and eSafe (phishing attack)**
 - Attacker impersonates and sets up a PSC and eSafe (not registered in TSL). Remark: eSafe does not need to be fully functional but only phish for credentials and generate subsequent connection error. (Optional: Attacker acquires a valid X509 certificate (buy one at Verisign))
 - Attacker pushes the fake PSC in Web search engine rankings
 - Honest SP selects fake PSC found e.g. by Google (establishes a HTTPS session) and believes to talk to a valid, official PSC
 - Attacker (PSC) provides honest SP with fake list of eSafes
 - SP chooses one of fake eSafes
 - Attacker impersonates eSafe and phishes SP's credentials
- **Result:**



- Attacker receives SP's eSafe credentials in impersonated eSafe.
- **Attack: Combined Phishing and Impersonation Attack**
 - Attacker impersonates a PSC and receives SP's eSafe credentials
 - Attacker impersonates a SP at PSC, using SP's eSafe credentials and finalizes a complete request.
- **Result:**
 - Attacker receives SP's eSafe credentials in impersonated eSafe.
 - Attacker can access or upload any data from or to the eSafe (e.g. access sensitive information/data, upload malware or illegal material)
 - Attacker completes eSafe scenario with stolen authentication credentials.
- **Reason:**
 - SP does not have the diligence or skills to verify the authenticity (e.g. certificates) of SPOCS components
- **Violates:**
 - Confidentiality
 - Accountability
 - Non-repudiation
- **Possible Risk Mitigation:**
 - Empowerment of a mandatory PSC specification that requires mandatory strong registration and strong authentication to SPs
 - Implementation of the combination of following measures:
 - Establish a unique naming convention for PSCs to provide an easy way for the ordinary user (SP) to recognize a valid PSC (like xyz.country.psc.gov.eu)
 - Setup a central European information website and a central directory
 - Provide the root (TSL) certificate for download on this site
 - Work with leading Web browser vendors to establish a built-in TSL verification
 - → quick win: establish TSL root certificate as factory default in leading Web browsers
 - Provide user education and public awareness campaigns



D.3.4 Risks due to a TSL Denial of Service Attack¹⁹

- **Attack: Denial of Service Attack against TSL Infrastructure**
 - Attacker waits until the current version TSL is about to expire
 - Attacker prevents the official TSL issuer to provide an updated version (e.g. via (D)DOS)
 - SPOCS components are not able to acquire new version of the TSL
 - After the expiry date, all specification compliant implementations must discard the outdated TSL
 - they do no longer have a valid TSL
 - they must discard all connection attempts, because they can no longer validate the TSL certificates presented
- **Result:**
 - All SPOCS components stop working!
- **Reason:**
 - SPOCS Components depend on the TSL being up-to-date for diverse cryptographic operations.
- **Violates:**
 - Availability
- **Risk mitigation:**
 - Conduct IT service continuity and disaster recovery planning, e.g. provide new TSL early enough to distribute it via alternate channels if necessary.
 - Set-Up of an overall Governance function to ensure and adequate management, accreditation and operation of the SPOCS TSL on the basis of The SPOCS D3.2 Appendix 5 SPOCS TSL Accreditation and Operation Policy.

¹⁹ Annotation: The TSL Denial of Service Attack is not a result of formal modelling and verification but has been revealed by additional security considerations made during the SPOCS project.



E. Appendix D – TSL Interface Code Usage Example

```
package eu.spocseu.tsl.test;

import eu.spocseu.tsl.Constants;
import eu.spocseu.tsl.TSL;
import eu.spocseu.tsl.TSLEntry;
import eu.spocseu.tsl.TrustedServiceImpl;
import eu.spocseu.tsl.internal.exception.TSLException;
import eu.spocseu.tsl.internal.exception.VerifyException;
import java.io.FileInputStream;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author D.M. - Infocert 2010
 * @version 0.1
 *
 * TSL interface initialization and searching example
 *
 */

public class Main {

    /**
     * @param args the command line arguments
     */
    @SuppressWarnings("static-access")
    public static void main(String[] args) {

        TSL tsl = new TSL("C:\\intProperties.xml");
```



```
try {
    tsl.init();
    ArrayList<TSLEntry> list = (ArrayList<TSLEntry>) tsl.getTSLEntries();
    for (TSLEntry t : list) {
        System.out.println("TSL country code: " + t.getCountryCode());
        System.out.println("TSL ID: " + t.getID());
        System.out.println("TSL scheme name: " +
t.getSchemeInformation().getSchemename());
        if (t.getSchemeOperatorCert() != null)
            System.out.println("TSL scheme operator cert subjetcDN: " +
t.getSchemeOperatorCert().getSubjectDN());
        System.out.println("TSL tsps: " + t.getTrustedServiceProvider().size());
    }
} catch (TSLEntryException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
} catch (VerifyException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}

try {
    System.out.println("*****NEW SEARCH*****");

    tsl.setsearchParameter_Name("*spocs-edelivery.eu");
    System.out.println("Search by name: " + tsl.getsearchParameter_Name());

    ArrayList<TrustedServiceImpl> servlist = (ArrayList<TrustedServiceImpl>)
tsl.startParametersSearch();

    System.out.println("Found services : " + servlist.size());

    int i = 0;
    for (TrustedServiceImpl t : servlist) {

        System.out.println("Service n. " + ++i);
        System.out.println("Service name: " + t.getServiceName());
    }
}
```



```
        System.out.println("Service type: " + t.getServiceType());
        System.out.println("Service cc: " + t.getTSLCountryCode());
        System.out.println("Service realmname: " + t.getRealmName());
        System.out.println("Service supply point: " +
t.getServiceSupplyPoints().toString());
        System.out.println("Service certificate subjectDN: " +
t.getDigitalId().getSubjectDN());
    }
    System.out.println("*****END SEARCH*****");
    System.out.println(" ");

    } catch (TSLException ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }

    try {
        System.out.println("*****NEW SEARCH*****");
        tsl.resetsearchParameter_All();

        tsl.setsearchParameter_Type(Constants.serviceType_eDelivery);
        System.out.println("Search by eDelivery serviceType: " +
tsl.getsearchParameter_Type());

        ArrayList<TrustedServiceImpl> servlist = (ArrayList<TrustedServiceImpl>)
tsl.startParametersSearch();

        System.out.println("Found services : " + servlist.size());
        int i = 0;
        for (TrustedServiceImpl t : servlist) {

            System.out.println("Service n. " + ++i);
            System.out.println("Service name: " + t.getServiceName());
            System.out.println("Service TSL country code: " + t.getTSLCountryCode());
            System.out.println("Service realmname: " + t.getRealmName());
            System.out.println("Service supply point: " +
t.getServiceSupplyPoints().toString());
```



```
        System.out.println("Service signature certificate: " +
t.getServiceSignatureCertificate().getSubjectDN());
        System.out.println("Service SSL certificate: " +
t.getServiceSSLCertificate().getSubjectDN());
        System.out.println(" ");
    }
    System.out.println("*****END SEARCH*****");
    System.out.println(" ");

    } catch (TSLException ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }

    try {
        System.out.println("*****NEW SEARCH*****");
        tsl.resetsearchParameter_All();

        tsl.setsearchParameter_Type(Constants.serviceType_eDelivery);
        tsl.setsearchParameter_CountryCode("DE");
        System.out.println("Search by eDelivery serviceType: " +
tsl.getsearchParameter_Type() +
        " and CountryCode: " + tsl.getsearchParameter_CountryCode());

        ArrayList<TrustedServiceImpl> servlist = (ArrayList<TrustedServiceImpl>)
tsl.startParametersSearch();

        System.out.println("Found services : " + servlist.size());
        int i = 0;
        for (TrustedServiceImpl t : servlist) {

            System.out.println("Service n. " + ++i);
            System.out.println("Service name: " + t.getServiceName());
            System.out.println("Service TSL country code: " + t.getTSLCountryCode());
            System.out.println("Service realmname: " + t.getRealmName());
            System.out.println("Service supply point: " +
t.getServiceSupplyPoints().toString());
```




```
        System.out.println("Service authLevels: " +
t.getAuthenticationLevel().toString());
        System.out.println("Service supportedEvidence: " +
t.getSupportedEvidence().toString());
        System.out.println("Service requestedEvidence: " +
t.getRequestedEvidence().toString());
        System.out.println("Service domains: " + t.getManagedDomain().toString());
        System.out.println("Service signature certificate: " +
t.getServiceSignatureCertificate().getSubjectDN());
        System.out.println("Service SSL certificate: " +
t.getServiceSignatureCertificate().getSubjectDN());
        System.out.println(" ");
    }
    System.out.println("*****END SEARCH*****");
    System.out.println(" ");

    } catch (TSLEException ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }

    try {
        System.out.println("*****NEW SEARCH*****");
        tsl.resetsearchParameter_All();

        ArrayList<String> parOpModes = new ArrayList<String>();
        parOpModes.add(Constants.eSafe_operationMode_PUSH);

        tsl.setsearchParameter_Type(Constants.serviceType_eSafe);
        tsl.setsearchParameter_CountryCode("AT");
        tsl.setsearchParameter_eSafeOperationMode(parOpModes);

        System.out.println("Search by eSafe serviceType: " + tsl.getsearchParameter_Type()
+ " and opModes " + parOpModes.toString() +
        " and CountryCode: " + tsl.getsearchParameter_CountryCode());

        ArrayList<TrustedServiceImpl> servlist = (ArrayList<TrustedServiceImpl>)
tsl.startParametersSearch();
```



```
System.out.println("Found services : " + servlist.size());
int i = 0;
for (TrustedServiceImpl t : servlist) {

    System.out.println("Service n. " + ++i);
    System.out.println("Service name: " + t.getServiceName());
    System.out.println("Service TSL country code: " + t.getTSLCountryCode());
    System.out.println("Service supply point: " +
t.getServiceSupplyPoints().toString());
    System.out.println("Service eSafe opModes: " + t.geteSafeOpModes().toString());
    System.out.println("Service eSafe CC: " + t.getExtensionCountryCode());
    System.out.println("Service certificate subjectDN: " +
t.getDigitalId().getSubjectDN());
    System.out.println(" ");
}
System.out.println("*****END SEARCH*****");
System.out.println(" ");
} catch (TSLException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}

try {
    System.out.println("*****NEW SEARCH*****");
    tsl.resetsearchParameter_All();

    tsl.setsearchParameter_Type(Constants.serviceType_PSC);
    tsl.setsearchParameter_CountryCode("AT");
    System.out.println("Search by PSC serviceType: " + tsl.getsearchParameter_Type() +
" and PSC countrycode " + tsl.getsearchParameter_CountryCode());

    ArrayList<TrustedServiceImpl> servlist = (ArrayList<TrustedServiceImpl>)
tsl.startParametersSearch();

    System.out.println("Found services : " + servlist.size());
```



```
int i = 0;
for (TrustedServiceImpl t : servlist) {

    System.out.println("Service n. " + ++i);
    System.out.println("Service name: " + t.getServiceName());
    System.out.println("Service type: " + t.getServiceType());
    System.out.println("Service TSL country code: " + t.getTSLCountryCode());
    System.out.println("Service supply point: " +
t.getServiceSupplyPoints().toString());
    System.out.println("Service certificate subjectDN: " +
t.getDigitalId().getSubjectDN());
    System.out.println(" ");
}
System.out.println("*****END SEARCH*****");
System.out.println(" ");
} catch (TSLException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}

try {
    System.out.println("*****NEW SEARCH*****");

    tsl.resetsearchParameter_All();

    TrustedServiceImpl t = tsl.getTrustedDomain("legalmail.it");
    System.out.println("Search by TrustedDomain name");

    X509Certificate cert = null;
    CertificateFactory cf = null;
    try {
        cf = CertificateFactory.getInstance("X.509");
        cert = (X509Certificate) cf.generateCertificate(new
FileInputStream("d:\\Documenti\\Progetti\\SPOCS\\test_mdaustria_signature.crt"));
    } catch (CertificateException ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```



```
    } catch (Exception ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }

    int i = 0;
    if (t != null) {

        System.out.println("Service n. " + ++i);
        System.out.println("Service name: " + t.getServiceName());
        System.out.println("Service type: " + t.getServiceType());
        System.out.println("Service TSL country code: " + t.getTSLCountryCode());
        System.out.println("Service supply point: " +
t.getServiceSupplyPoints().toString());
        System.out.println("Service certificate subjectDN: " +
t.getDigitalId().getSubjectDN());
        System.out.println("Service verified: " + tsl.verifyDomain("legalmail.it",
cert));

        System.out.println(" ");
    } else
        System.out.println("Service null");
        System.out.println(" ");
        System.out.println("*****END SEARCH*****");
        System.out.println(" ");
    } catch (TSLException ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }

}

}
```