## Abstract

This document describes the final version of the Commius architecture. We have finalized the scope of the Commius software architecture and defined the whole conceptual architecture for Commius. The conceptual architecture provides an overall view of system design and identifies major technical concerns for Commius to ensure that the development of Commius will achieve the goals of interoperability service utility. The Commius architecture is designed as an open, secured and customizable system, supporting networks of cooperative SMEs to perform their daily business based on emails.
# Table of Contents

**TABLE OF CONTENTS**............................................................................................................. 2  
**LIST OF FIGURE** .................................................................................................................. 4  
**ABBREVIATIONS** .................................................................................................................. 5  

1  **INTRODUCTION** ................................................................................................................. 6  
  1.1  SCOPE OF THE DOCUMENT ............................................................................................... 6  
  1.2  APPLICABLE AND REFERENCE DOCUMENTS ................................................................... 6  
  1.3  REVISION HISTORY ............................................................................................................. 8  

2  **EXECUTIVE SUMMARY** ...................................................................................................... 11  

3  **THE COMMIIUS INTEROPERABILITY SERVICE UTILITY** .................................................. 12  
  3.1  PROPERTIES OF THE COMMIIUS ISU .............................................................................. 12  
  3.2  THE BENEFIT OF COMMIIUS ISU .................................................................................... 14  

4  **TERMINOLOGIES** .............................................................................................................. 16  
  4.1  SOFTWARE COMPONENT .................................................................................................. 16  
  4.2  SOFTWARE MODULE ....................................................................................................... 16  
  4.3  SOFTWARE SERVICE ....................................................................................................... 17  
  4.4  COMMIIUS SERVICE UTILITY (CSU) ............................................................................. 17  
  4.5  PROCESS TYPE AND INSTANCE ....................................................................................... 17  
  4.6  TEMPLATES .................................................................................................................... 18  

5  **STATE OF THE ART** ............................................................................................................ 19  

6  **REQUIREMENTS FOR COMMIIUS SOFTWARE ARCHITECTURE** ............................... 22  

7  **COMMIIUS ARCHITECTURAL OVERVIEW** .................................................................... 25  
  7.1  OVERVIEW ..................................................................................................................... 25  
  7.2  OVERVIEW OF EMAILS HANDLING PROCESS ............................................................... 27  
  7.3  EXAMPLE OF COMMIIUS USAGE: AN ILLUSTRATIVE PARTNER SEARCH SCENARIO 27  

8  **DETAILED DESIGN OF THE COMMIIUS ARCHITECTURE** ............................................. 30  
  8.1  USER TOOLS ................................................................................................................... 30  
  8.1.1  Tools for assisting end-users to perform their business ................................................. 30  
  8.1.2  Tools for managing and customizing COMMIIUS .......................................................... 30  
  8.2  EMAIL GATEWAY PLUGINS ............................................................................................ 31  
  8.2.1  Message Post Processing component ......................................................................... 32  
  8.3  COMMIIUS MODULES .................................................................................................... 33  
  8.3.1  System-oriented modules ............................................................................................. 33  
  8.3.1.1  Authentication/Authorization .................................................................................... 33  
  8.3.2  Business modules ........................................................................................................ 34  
  8.4  COMMIIUS MODULE MANAGEMENT ........................................................................... 35  
  8.4.1  Commiius Module Manager ......................................................................................... 35  
  8.4.2  COMMIIUS Module Discovery .................................................................................... 36  
  8.5  SYSTEM INTEROPERABILITY .......................................................................................... 36  
  8.5.1  System Interoperability Components ............................................................................ 36  
  8.5.1.1  Message Decomposition .......................................................................................... 37  
  8.5.1.2  Information Extraction .............................................................................................. 37  
  8.5.1.3  System Connectors .................................................................................................... 37  
  8.5.2  System Interoperability Control and Data Flows .......................................................... 39  
  8.6  SEMANTIC INTEROPERABILITY ....................................................................................... 39  
  8.6.1  Semantic Interoperability Components ........................................................................ 40  
  8.6.1.1  Semantic Manager .................................................................................................... 40  
  8.6.1.2  Semantic Annotator .................................................................................................. 40  
  8.6.1.3  Document Interpreter .............................................................................................. 41  
  8.6.1.4  Semantic Negotiator .................................................................................................. 41  
  8.6.1.5  Document Type Manager .......................................................................................... 41
8.6.2 Semantic Interoperability Control and Data Flows ........................................ 42
8.7 PROCESS INTEROPERABILITY ............................................................................. 44
  8.7.1 Process Interoperability Components .............................................................. 45
    8.7.1.1 Enterprise Process Repository ................................................................. 45
    8.7.1.2 Process Instance Database ..................................................................... 45
    8.7.1.3 Detecting .................................................................................................. 46
    8.7.1.4 Tracking .................................................................................................. 46
    8.7.1.5 Assisting .................................................................................................. 46
    8.7.1.6 Advising .................................................................................................. 46
  8.7.2 Process Interoperability Control and Data Flows ......................................... 46
  8.7.3 The Interorganizational flow ......................................................................... 48
8.8 DATA MANAGEMENT COMPONENTS ................................................................ 49
8.9 HOSTING ENVIRONMENT .................................................................................... 50
8.10 EXAMPLE OF A BUSINESS MODULE ............................................................. 50
9 SECURITY AND PRIVACY CONCERNS ................................................................. 53
  9.1 SECURITY CONCERNS ..................................................................................... 53
  9.2 SECURITY AND PRIVACY SOLUTIONS IN COMMIS ........................................ 55
    9.2.1 Confidential Information Concerning Privacy Policies ............................... 56
    9.2.2 Authentication of the user ......................................................................... 59
    9.2.3 User Tools and EMail Gateway Plug-ins .................................................... 60
    9.2.4 Security Connectors .................................................................................... 60
    9.2.5 External security policies ............................................................................ 61
  9.3 SECURITY COMPONENTS .................................................................................. 61
    9.3.1 Third Party Security Components ................................................................ 62
    9.3.2 Commis-developed Security Components ................................................. 63
      9.3.2.1 Filters .................................................................................................. 63
      9.3.2.2 Encrypted persistence .......................................................................... 63
      9.3.2.3 User Management API ........................................................................... 63
      9.3.2.4 Security Classification .......................................................................... 63
  9.4 SECURITY/PRIVACY CONTROL AND DATA Flows .................................... 64
10 EXTERNAL SYSTEMS ............................................................................................ 67
11 CUSTOMIZATIONS AND DEPLOYMENTS .......................................................... 69
  11.1 INDIVIDUAL COMMIS ..................................................................................... 69
  11.2 NETWORK OF COOPERATIVE SMEs .............................................................. 69
    11.2.1 Centralized Shared Resource Systems ....................................................... 70
    11.2.2 Peer-to-Peer Shared Documents ............................................................... 71
    11.2.3 Peer-to-Peer Model for Cooperative Activities ......................................... 72
  11.3 COMMIS SERVICE PROVIDERS ..................................................................... 73
  11.4 CLIENT-SIDE EXTENSION ............................................................................. 74
12 IMPLEMENTATION ROADMAP ............................................................................. 76
  12.1 ANALYSIS OF EXISTING TECHNOLOGIES FOR NETWORKED ENTERPRISES ...... 76
    12.1.1 Enterprise Service Bus (ESB) .................................................................. 76
    12.1.2 Web Services ......................................................................................... 76
    12.1.3 Service Composition Architecture (SCA) ............................................... 77
    12.1.4 OSGi ...................................................................................................... 77
  12.2 COMMIS HOSTING ENVIRONMENT AND PROGRAMMING MODELS ........ 77
  12.3 THE ROADMAP FOR THE IMPLEMENTATION OF COMMIS ARCHITECTURE ........ 78
13 CONCLUSIONS ...................................................................................................... 79
List of Figure

Figure 1: Deliverables dependencies diagram .......................................................... 7
Figure 2: The Commius Interoperability Service Utility ............................................. 13
Figure 3: The Commius general workflow ................................................................ 22
Figure 4: Architectural requirements based on the Commius Workflow ..................... 24
Figure 5: Architectural overview of Commius ............................................................. 25
Figure 6: Overview of how Commius handles emails ................................................. 27
Figure 7: Search partner scenario ............................................................................. 29
Figure 8: Design of a Mail Gateway Plugin for Postfix MTA ....................................... 31
Figure 9: Component view of Email Gateway Plugin .................................................. 32
Figure 10: MPP component view ............................................................................. 32
Figure 11: Module Manager and dependent repositories ............................................. 35
Figure 12: Component view of system interoperability .............................................. 36
Figure 13: Component diagram of System Connector ............................................... 38
Figure 14: System Interoperability Control and Data flows ....................................... 39
Figure 15: Components of the Semantic Interoperability Layer .................................. 41
Figure 16: An example of a flow illustrating the use of Semantic Interoperability Components ................................................................................................................................. 43
Figure 17: Configuration of the In-House process database ....................................... 45
Figure 18: Component view of Process Interoperability ............................................. 45
Figure 19: Process Interoperability control flow ......................................................... 47
Figure 20: Interorganizational use case example of Commius .................................... 49
Figure 21: (Simplified) sequence of the Order Receiving and Enriching Module ....... 52
Figure 22: Security concerns and solutions in Commius ............................................ 56
Figure 23: Interaction between security and semantic layers ..................................... 57
Figure 24: Sequence diagram for the interaction between security and semantic layers .. 57
Figure 25: Commius privacy concerns ..................................................................... 58
Figure 26: (Complete) Sequence of email security actions ....................................... 59
Figure 27: Overview of security components ............................................................. 61
Figure 28: A general sequence of filtered invocations ............................................... 64
Figure 29: Flow of a chain of filters ......................................................................... 65
Figure 30: A general sequence of access to the encrypted persistence component ....... 66
Figure 31: A network of cooperative Commius-based SMEs ...................................... 70
Figure 32: Simplified example of storing and requested shared documents ............... 71
Figure 33: A simplified example of a cooperative activity in answering a request ......... 73
Figure 34: Server-side and client-side models ............................................................ 74
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>Authentication and Authorization</td>
</tr>
<tr>
<td>CSP</td>
<td>Commius Service Provider</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>ESB</td>
<td>Enterprise Service Bus</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>ISU</td>
<td>Interoperability Service Utility</td>
</tr>
<tr>
<td>MPP</td>
<td>Message Post Processing</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source Software</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-Peer</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software-as-a-Service</td>
</tr>
<tr>
<td>SIL</td>
<td>Semantic Interoperability Layer</td>
</tr>
<tr>
<td>SME</td>
<td>Small and Medium Enterprise</td>
</tr>
<tr>
<td>WP</td>
<td>Workpackage</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Scope of the document

This document describes the final version of the Commius architecture, constituting the deliverable “D3.1.2: Commius architecture”. It identifies the final scope of the Commius software architecture and defines the conceptual architecture for Commius. The conceptual architecture provides an overall view of system design and identifies major technical concerns for Commius to ensure that the development of Commius will achieve the goals of interoperability service utility. This deliverable is a continuous work of the first version of the Commius architecture issued at M6 [D3.1.1].

This deliverable has revised all major points in the first version. In addition, the following issues have been addressed in this deliverable:

- The execution model of the module manager
- The scenario of using components across Commius layers
- How to support different types of service providers
- Privacy identification and recommendation
- Network of cooperative SMEs and customization and deployment of Commius
- The implementation platform

The revision and new work in this deliverable have been made based on the feedbacks obtained from the prototype development and user requirements.

Out of the scope of this document is:

- The detailed design of system, semantic and interoperability layers
- Detailed specification of business modules and module managements
- Detailed customization and deployment techniques

These issues will be addressed in the detailed design of Commius techniques and software prototypes in appropriate WPs, such as in WP4, WP5, WP6 and WP7. Furthermore, the detailed state of the art related to Commius architecture given in a separate document named “Appendix to D3.1: State of the Art in Interoperability Technologies” [D3.1-SoTA] has been updated, serving as an appendix for this deliverable.

1.2 Applicable and reference documents

This deliverable depends on the initial version of the Commius architecture, described in the deliverable D3.1.1 [D3.1.1] and the initial version and final version of the pilot scenarios requirements, described in the deliverables D2.1.1 [D2.1.1] and D2.1.2 [D2.1.2], respectively. Furthermore, several other deliverables are dependent on this deliverable. Figure 1 describes the dependencies between this deliverable and other deliverables.
This deliverable refers to the following documents:


[D2.1.2] Kalaboukas Konstantinos et al., D2.1.2 – Pilot Scenario Requirements (final version), Commius Deliverable, Jan 2009.


[SM06] Iain Duncan Stalker, Nikolay Mehandjiev; A Devolved Ontology Model for the PragmaticWeb, presented at the first international conference on the pragmatic web, ICPW 2006. Best Paper Award.


### 1.3 Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>17-03-2008</td>
<td>Hong-Linh Truong</td>
<td>Initial document</td>
</tr>
<tr>
<td></td>
<td>21-03-2008</td>
<td>Hong-Linh Truong</td>
<td>Initial requirement from user stories</td>
</tr>
<tr>
<td></td>
<td>07-04-2008</td>
<td>Hong-Linh Truong, Marek Cigan, Enrico Morten, Michal Laclavik, Martin Seleng, Emil Gatial, Ian</td>
<td>Update terminologies, mail plug-ins, connectors</td>
</tr>
<tr>
<td>0.2</td>
<td>18-04-</td>
<td>Hong-Linh Truong, Silke</td>
<td>Update process</td>
</tr>
</tbody>
</table>

© Commius Consortium
<table>
<thead>
<tr>
<th>Date</th>
<th>Authors</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>Balzert, Thomas Burkhart, Dirk Werth, David Rodriguez Nieto, Iain D. Stalker</td>
<td>interoperability, security, semantic interoperability</td>
</tr>
<tr>
<td>0.3</td>
<td>2-05-2008 Hong-Linh Truong, Iain D. Stalker</td>
<td>Update semantic interoperability and examples</td>
</tr>
<tr>
<td>06-05-2008</td>
<td>Michal Laclavik, Marian Babik, Marek Ciglan</td>
<td>Various updates, system interoperability, semantic interoperability</td>
</tr>
<tr>
<td>07-05-2008</td>
<td>Martin Seleng, Emil Gatial, Zoltan Balogh</td>
<td>Updates to system interoperability, added one new chapter Message Post Processing and initial technical tools and solutions</td>
</tr>
<tr>
<td>12-05-2008</td>
<td>Emil Gatial</td>
<td>Updates to system interoperability, server and client side point of view of Message Post Processing architecture and also of overall WP4 architecture</td>
</tr>
<tr>
<td>0.4</td>
<td>2-06-2008 Paolo Gianrossi, Martin Carpenter, Thomas Burkhart, Hong-Linh Truong</td>
<td>Update all</td>
</tr>
<tr>
<td>0.5</td>
<td>27-06-2008 Hong-Linh Truong, Martin Carpenter, Enrico Morten, Michal Laclavik, Kalaboukas Konstantinos, Panagiotis Gkouvas</td>
<td>All sections</td>
</tr>
<tr>
<td>0.6</td>
<td>10-07-2008 Hong-Linh Truong, Enrico Morten, Michal Laclavik, Martin Šeleng, Emil Gatial, Kalaboukas Konstantinos, Thomas Burkhart, Dirk Werth, Dario Luiz Lopez</td>
<td>All sections</td>
</tr>
<tr>
<td>0.7</td>
<td>16-07-2008 Hong-Linh Truong, Kalaboukas Konstantinos, Thomas Burkhart, Dario Luiz Lopez, Martin Šeleng</td>
<td>All sections</td>
</tr>
<tr>
<td>0.8</td>
<td>25-07-2008 Hong-Linh Truong, Dario Luiz Lopez, Enrico Morten</td>
<td>Update ISU, security</td>
</tr>
<tr>
<td>0.9</td>
<td>28-07-2008 Hong-Linh Truong</td>
<td>Update all based on review comments</td>
</tr>
<tr>
<td>1.0</td>
<td>30-07-2008 Enrico Morten</td>
<td>Final editing</td>
</tr>
</tbody>
</table>
| 1.1        | 12-12-2008 Hong-Linh Truong                                            | Update all sections based on agreement in                                                                                                                                
<table>
<thead>
<tr>
<th></th>
<th>Date</th>
<th>Participant(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>16-12-2008</td>
<td>Dario Luiz Lopez</td>
<td>Privacy issues</td>
</tr>
<tr>
<td>1.3</td>
<td>15-01-2009</td>
<td>Hong-Linh Truong, Enrico Morten</td>
<td>Service provider model, network of Commius</td>
</tr>
<tr>
<td>1.4</td>
<td>18-01-2009</td>
<td>Hong-Linh Truong</td>
<td>Update all sections</td>
</tr>
<tr>
<td>1.5</td>
<td>28-01-2009</td>
<td>Hong-Linh Truong, Cesar Marin</td>
<td>Update semantic interoperability</td>
</tr>
<tr>
<td>1.6</td>
<td>30-01-2009</td>
<td>Hong-Linh Truong, Giovanni Casella, Thomas Burkhart</td>
<td>Update process interoperability, implement review comments</td>
</tr>
<tr>
<td>2.0</td>
<td>30-01-2009</td>
<td>Hong-Linh Truong</td>
<td>Final check, deliverable dependency diagram</td>
</tr>
</tbody>
</table>
2 Executive summary

Today's individual, specialized SMEs (small and medium enterprises) are seeking solutions that enable them to cooperate with each other and to establish a network of cooperative SMEs through which they can offer complementary services to be able to stay competitive. Being able to perform cooperative business with other SMEs is crucial to an SME but there are many obstacles preventing SMEs doing so. For example, within (cooperative) SMEs, different types of documents and requests are exchanged, diverse types of flexible and adaptive business processes are executed, and various legacy systems need to be supported; all of them are related to the interoperability among SMEs at different levels, such as software system, semantic documents, and business process. However, SMEs cannot afford to have skilled IT to manage the complex network of SMEs or to take a long learning curve to master complex software. SMEs need simple, almost zero-cost solutions.

Motivated by the lack of almost zero-cost and simple, yet powerful, solutions for SMEs to deal with SME interoperability and cooperation, the Commius project aims at providing a utility-like capability for SMEs that is accessible to all enterprises at a very low cost. Such a capability will guarantee a certain set of common rules for doing businesses and is not owned by any single entity. To this end, the Commius project relies on the concept of supporting enterprises doing their business via emails by providing open source software. As most SMEs rely on emails for conducting their businesses, email-based solutions will be open, flexible and Internet-scale.

To realize the Commius vision, the Commius conceptual architecture is designed to provide a generic software framework which is not only extensible and modular but also easy to use. We have identified major aspects in the mapping from the ISU (Interoperability Service Utility) vision into the context of the Commius objectives and determined basic requirements for the Commius software. This deliverable presents the final version of the Commius conceptual architecture which includes (i) the Commius Email-based Interoperability Service Utility model, (ii) the overall architecture describing various software components and modules, (iii) security and privacy issues, (iv) Commius external systems, and (v) the implementation roadmap.

The Commius architecture is designed as an open, secured and customizable system, supporting networks of cooperative SMEs to perform their daily business based on emails.
3 The Commius Interoperability Service Utility

Following the suggestion in the IST Enterprise Interoperability Research Roadmap¹, the Commius project conceives interoperability as a utility-like capability (Interoperability Service Utility – ISU) for enterprises:

1. Accessible in principle by all enterprises (universal or near-universal access)
2. Available at (very) low cost, with zero start-up costs in terms of both resources and learning;
3. “Guaranteed” to a certain extent and at a certain level in accordance with a set of common rules
4. Not controlled or owned by any single private entity

The Commius conceptual architecture aims at providing a high-level view and design of software for achieving the above-mentioned concepts. Therefore, it is necessary to map these concepts to the context of the Commius architecture and to explain how the user can benefit from the Commius ISU.

3.1 Properties of the Commius ISU

Figure 2 describes the concrete technical mapping from the ISU vision to the Commius utility. The mapping provides the basic requirements for the design and development of the Commius architecture:

1. Accessible in principle by all enterprises: By “accessible” we mean that Commius will be available to all enterprises which use email to carry on their business. Commius will support the development of user-centric interoperability services on top of email communication suitable to different business models, such as simple desktop installation, mail server installation in company premises, Commius services through an external service provider, and outsourcing Commius installation and maintenance.

2. Scalable and almost zero cost: SMEs business is mainly carried on through emails and email communication is Internet-scale and is almost zero cost. Commius supports interoperability solutions for (cooperative) business activities performed and documents exchanged via email (in human readable form). Commius will be easy-to-use and impose no modification to the usual SME practices: companies without Commius can simply download (free) Commius modules that will enable the companies to utilize Commius interoperability solutions. Commius can also be deployed to a full interoperability system without the need to invest and install complex infrastructure and software, such as through service providers, following the concept of SaaS (Software-as-a-Service)². The management and usability will be addressed at all levels: end-user, administrator and developer.

3. Guaranteed: Commius guarantees the quality by the reuse of the achievements of the other research projects, by the adoption of the available standards and by its approach, based on simple services always available, such as Web and email. Moreover, it will address security and privacy issues to ensure that business can be carried out in a trusted manner.

² http://en.wikipedia.org/wiki/Software_as_a_Service
4. **Not controlled by single entity:** By “controlled” we mean that the software produced by the Commius will not be owned by a single entity (e.g., a company or an organization). The core of Commius will be released under an OSS (Open Source Software) license. The project will foster the creation of an OSS community which will use the Commius core to develop Commius modules dedicated to specific interoperability business application. Such modules may be developed by software providers who will adopt either open source or proprietary business approach.

5. **Supporting a network of cooperative enterprises:** The bottom-up and evolutionary approach of Commius will enable creative cooperation among SMEs. SMEs with different skills and cultural backgrounds will gradually approach the interoperability and cooperate with other SMEs in order to create and diffuse interoperability services suitable for their business models. SMEs will become aware of the opportunities created by ICT and will be encouraged to adopt business models based on innovation and global collaboration.

---

**Figure 2: The Commius Interoperability Service Utility**

- **Plug in email utilities**
  - Interface through mail client and Web browser
  - Allows working “as usual”

- **Easy to use**
  - Based on mail messages and Web interface
  - Perceived as an assistant
  - Simple configuration phase

- **User centric**
  - Focussed and driven by concrete real SME needs
  - Open to all the business models and existing technologies

- **Guaranteed services**
  - Adopt standards
  - Exploits results of other projects
  - Based on always available services
  - Ensure security and privacy

- **Adaptable**
  - Gradually adaptable to the growing needs of the SMEs
  - CustomisD4.1.1 Interoperability over SMTP (initial version)

- **Support a network of cooperative enterprises**
  - Bottom-up and evolutionary approach
  - Support creative collaboration among SMEs
  - Encourage Business Models based on innovation and global collaboration
3.2 The Benefit of Commius ISU

According to the Commius vision of the ISU the following reference scenario of deployment and use of the expected outcomes of the project is envisioned:

An SME is not using any Interoperability tool but is facing the problem of improving its competitiveness by adopting business models and strengthening its cooperation with other enterprises.

Commius dissemination is effective. The SME knows about Commius, which claims an almost zero-cost and easy-to-use access to Interoperability.

The SME visits the Commius community portal and decides to download and install the Commius plug-in for its email system. No special skill or knowledge is required, no costs will be charged, the Commius “first step” is installed. The SME carries on its business as usual.

Commius, using the default configuration, starts monitoring the emails flow. Commius recognizes basic documents, such as Request for Quotation, Purchase Order, and Invoices, keeps tracks of the workflow in its repository, and starts supporting the SME’s work by suggesting simple actions.

The SME finds that the Commius suggestions are useful. Then, it decides to dedicate some time to configure Commius. The SME defines and describes the templates of its business documents, its business models, the main workflows, and the security policy.

The SME receives documents and emails from other SMEs which have installed Commius. Interoperability tasks are enabled. Searching for partners, purchases, ordering and invoicing are monitored by Commius, which suggests the right actions and prepares the right documents according to the SME’s workflow.

Commius’s offers satisfy the SME. Thus, the SME decides that it is worth requiring extra services. The SME contacts the Commius Community and asks who is available for doing the customisation. The SME finds out some IT companies located not far from the SME. The SME contacts the IT companies and negotiates and agrees about the customisation work required.

The chosen IT company has completed the customisation work. Now Commius is able to access directly the SME legacy systems used for ERP and can automate some business actions.

The SME is contacted by a new enterprise: it is required to supply components in the delivery of a complex system where several SMEs are involved. The new enterprise sets up a dedicated Commius network to support Interoperability within the specific...
workflow. The SME is now a node of the network and can interoperate with the other enterprises.

The Commius network has subscribed, with a Commius service provider, an external service for document standard and workflow models maintenance and update. Other, more complex, services are available for the Commius users, such as financial and market information, translations, legal information, and a list of contacts in different business sectors.

ERP software vendors develop Commius Modules to support interoperability within their platforms. Ebusiness providers make available Commius modules to support interoperability with their clients.
4 Terminologies

We have described the Commius ISU concept and its implication for the development of the Commius architecture. This section aims at providing basic terminologies which will be used to describe technical components of the architecture.

4.1 Software Component

A software component is well-defined and understood in literature. In Commius, a software component implements a predefined functionality and it adheres an interface through which the functionality can be utilized. A Commius software component is the fundamental unit in the Commius architecture. In this deliverable, we only describe the functionality of main Commius software components but not how they are implemented in detail.

Examples of Commius software components:

- Any external/legacy system connector
- Any information extraction or processing component
- Any semantic annotation or tagging component
- System connector to SQL databases
- Identification of semantic concepts and individuals in communication
- Pattern based information extraction

4.2 Software module

In Commius, a software module is a collection of software components that provides a predefined functionality. All software modules will follow a predefined interface and consist of code and configuration data. Software components are composed into a module, based on their functionalities, suitable for different purposes. A software component may be packaged into different software modules. Software module provides generic customizable or concrete interoperability functionality.

We distinguish two types of software module: system module and business module. A system module provides basic functionalities to ensure the operation of Commius-enable systems and to support the development of business solutions atop Commius systems. A Commius business module offers a set of functionalities related to a particular business or interoperability goal. A Commius business module can be exploited in several business processes and can support a different internal configuration for each business process in which it is exploited. The Commius system will enable the user to add/update or substitute business modules in order to customize or add new functionalities to the system. Commius business modules will be able to exploit external services and components developed. Several generic interoperability modules will be developed to realize interoperability needs of Commius pilot business cases. Third parties outside of Commius consortium will be able to deliver new Commius modules based on defined API/interfaces. Such modules can wrap-up or adopt existing interoperability solutions to Commius or deliver new interoperability functionality.
Users of Commius can discover and install new modules from various providers and thus realize adaptability of Commius. Examples of Commius software modules are:

- Module for providing information about the enterprise from public and government registers.
- Module for showing all info about the customer from intranet system, e.g. invoices payments.
- Module for processing invoices in ebXML

### 4.3 Software Service

Though the definition of a software service has different versions, it is well understood that a software service offers concrete functionalities that can be accessed through the network based on well-defined interface\(^3\). In the context of Commius, a software service is understood as a networked, deployable and runnable software application. The application logic of a software service can include multiple software components and modules, and/or can compose other software services. A Commius module can also be exposed as a service. As a runnable application, a software service might have necessary data in order to operate and provide its functionality. An example of a software service in Commius is a service to support remote management of user profiles that is built based on Authentication/Authorization module.

### 4.4 Commius Service Utility (CSU)

The Commius service utility (CSU) is a reference implementation of an Interoperability Service Utility (ISU). The Commius service utility includes services, which include modules (which include components) - to offer functionalities for Enterprise Interoperability. From the technical point of view, a Commius service utility (i) provides interoperable interface and concrete functionalities, (ii) offers functionalities addressing Enterprise Interoperability, and (iii) operates as an independent system.

The Commius utility supports the definition of ISU in the context of the Commius project, as given in Section 3. Given the Commius service utility, end-user tools/applications can be developed to utilize the utilities. Hence, we should distinguish between the utility and the end-user tools/applications utilizing the utility.

The Commius service utility will be deployed and operate on an infrastructure which, in Commius context, is based on email systems.

### 4.5 Process Type and Instance

In Commius, a process type consists of meta-process-knowledge, which is executed identically every time triggered, based on a predefined reference process template. A process instance is a specific occurrence of a process type.

---

\(^3\) [http://www.w3.org/TR/2004/NOTE-ws-arch20040211/]
4.6 Templates

We distinguish two types of templates: document templates and business process templates. A document template (or document model template), such as invoices, describes a common template model facilitating the semantic interoperability and automatic integration of various enterprise applications. A business process template (process model template) specifies a common sequence of steps of a particular business workflow.
5 State of the Art

Commius will provide ISU solutions based on email systems for SMEs. It will address various issues related to system, semantic and process interoperability atop email systems. In the following, we will summarize state of the art relevant to the Commius architecture. A detailed state of the art relevant to the Commius project is provided in an extra report as an appendix for the deliverable D3.1.2. Therefore, for the detailed state of the art, the report named “Appendix to D3.1- State of the Art in Interoperability Technologies” [D3.1-SoTA] should be referred. Furthermore, related work to Commius can also be found in [DoW].

Although, from a technical perspective, SME interoperability could be best achieved using a HTTP based solution, the Commius approach is based on email systems. The rationale is that email systems, based on SMTP and with diverse email clients available, are widely used in most SMEs. Furthermore, email communication is of asynchronous nature and does not require specialized systems which are 24 hours online. It can be processed using a mixture of automated or manual process steps and still is human readable. Various projects have connected email to knowledge or context-sensitive information, such as kMail4 and Zimbra5. Additionally, extensive work on email processing and active context-sensitive information and knowledge provision has been developed in ACoMA [LSH07] and EMBET frameworks. Another major research direction is to address task management in emails, such as visualization of complex conversations [WRGK05], recognition of relations between messages [BDHS03], social profiles and network [WJN+04], users’ email behaviour[HJH99], and filing messages [SK99]. There are also various commercial efforts atop email for business, such as collective knowledge gathering in MarkMail6, and prioritizing emails in Attent solution from Seriosity7. However there are various problems with the above-mentioned research efforts, when considering these efforts for SMEs. One major problem is that many of these systems involved the development of new standalone clients – forcing people to switch to new software and change their work practices. A second problem is that these systems were often not well integrated with users’ external/legacy systems (e.g., their file system or company databases) which meant that information in these systems could not be exploited or leveraged to allow more effective information processing.

Recently, there is also ongoing research on extracting information and knowledge, its management, social network analysis, and topic discovery from emails, such as in the introduction of the Enterprise track within the series of TREC conferences in 20058 and the Nepomuk project – Semanta9. However, supports of business interoperability as what Commius intends for SMEs are missing.

With respect to semantic interoperability, currently, semantic interoperability is typically achieved “off-line”, mainly through adherence to standards governing both the structure of the information being exchanged and the manner in which this information should be processed (for example, RosettaNet10 and ebXML11); or in particular, through commitment to

4 http://kontact.kde.org/kmail/
5 http://www.zimbra.com/
6 http://markmail.org
7 http://www.seriosity.com/products.html
9 http://smile.deri.ie/projects.html#semantice-mail
10 http://www.rosettanet.org/cms/sites/RosettaNet/
a shared ontology\textsuperscript{12}, which is then used as metadata. Several annotation protocols exist, including Annotea\textsuperscript{13}, Rubby\textsuperscript{14} and RDF annotation\textsuperscript{15}. Annotation solutions [U+05] can be manual, such as CREAM and Magpie, or semi-automatic based on natural language processing (NLP), a document structure analysis or a learning approach (which requires training sets or supervision). Moreover, there exist pattern-based semi-automatic solutions, such as PANKOW and C-PANKOW [CLS05], SemTag or pattern based approach Onteal[L+07].

The creation of domain specific ontologies and in particular as extensions of agreed, "upper" ontologies achieves semantic interoperability through semantic agreement but one must "learn" the local structures and create appropriate mapping rules or "meta-structures". This is clearly not "zero-cost" and is not suitable for SMEs. An alternative approach to this problem has been investigated as part of the MaBE and Crosswork European research projects. It advocates abstracting from matters of syntax to focus on the information passed within a concept. When a concept is passed to a recipient with a local ontology, the approach allows a range of options, starting from only retaining the subset of the concepts overall information which that recipient will understand; to a decision by the recipient to learn the complete new concept because of the future utility of this concept. This permits actors/trading partners to meaningfully communicate without the need for potentially expensive ontology agreement or alignment. A fuller account of this approach can be found in [SM06]. As observed in this section, current solutions in this area are not applicable for Commius due to high cost, which are not affordable for SMEs. Hence, the Commius project will follow a new approach which concentrates more on the information passed within a concept, than focusing on the syntax of information.

Various process interoperability techniques have been developed. With respect to Business Process Modelling for Collaborative Processes, within the EU-Funded project eJustice a comprehensive visualisation of legal processes has been realised. The experiences and results of this project can be used within Commius to support a business driven visualisation of collaborative processes and therefore allow comprehensive knowledge management for all partners involved. Within the research projects ATHENA and INTEROP, interoperability of enterprise systems at the business process level was analysed. The goal of these projects was to provide a solution that enables the modelling and execution of Cross-Organisational Business Processes (CBPs) [KVZ+06]. These solutions will also be extended in Commius to ensure that they fully address the requirements of SMEs. Modelling approaches include the Zachman Framework for Enterprise Architecture, GERAM, CIMOSA, ARIS and others. Frameworks for integrating enterprise applications include MISSION and others. There are some industry initiatives which are working on enterprise modelling concepts (standards), such as the Business Process Management Initiative (BPMI), the Workflow Management Coalition (WFMC), Open Applications Group, Inc (OAGIS GIS), Business Process Execution Language (OASIS BPEL), Business-Centric Methodology (OASIS BCM), Microsoft's BizTalk Server (Biztalk), Rossetta Net, W3C, OMG and many more (for details see http://www.athena-ip.org/). Various enterprise modelling languages have been developed to

\begin{thebibliography}{\textsuperscript{11}}
\bibitem{11} http://www.ebxml.org/
\bibitem{13} http://www.w3.org/2001/Annotea/
\bibitem{14} http://www.w3.org/TR/ruby/
\bibitem{15} http://iflt.org/discovery/2001/04/annotations/
\end{thebibliography}
support the building of a model of an enterprise according to different views, (function, process decision, economic, etc.) in an integrated way, such as EN/ISO 19440, ODP, UEML and GRAI, ebXML, WS-BPEL, and the event-driven process chain (EPC). In particular, EPC is one of the most popular and widely-accepted modelling methods for business processes. Within several projects the use of the EPC for the visualisation of inter-organizational processes has been proven (e.g. ATHENA, Interop and ArKoS). An extension of this modeling method to satisfy SMEs requirements within collaborative business scenarios has been elaborated within these projects and will be of value to Commius. A number of tools supporting the different modeling approaches and paradigms are available on the market, in many different varieties regarding features sets, sizes and costs\textsuperscript{16}. However, most are either too complex requiring a long learning curve, or are simply too expensive to be considered appropriate for SME environments. The lack of applicable modeling tools for SMEs explains the non-existent or insufficient level of formalization of SMEs’ internal and collaborative business processes. Such formal definitions of business processes are, however, fundamental to a full understanding and analysis of an SME’s strategic and operative impact on its objectives, and even more for implementing them as business processes. Even though much effort have been done in process interoperability, hardly none of the present approaches particularly matches the exigencies of SMEs.

In overall, we conclude that existing works provide various interoperability techniques in different frameworks. However, most techniques and frameworks in the field of interoperability are not tailored for SMEs. The Commius will reuse existing works in the sense that it will extract and tailor existing results for SMEs. A detailed analysis of related projects and the possibility of reusing existing results are provided in the appendix to this deliverable [D3.1.1-SoTA]. Built upon tailored existing results, The Commius will provide new solutions and support SMEs by providing a framework specially made for SMEs.

\textsuperscript{16}ATHENA (FP6 IST – 507849) DA.1.1.1 – State of the Art in EM Techniques and Technologies to Support Enterprise Interoperability
6 Requirements for Commius Software Architecture

Requirements and use cases for the Commius system have been derived from the active participation and collaboration of the users with the technical partners and are presented in detail in Deliverables D2.1 Pilot Scenarios Requirements (M6 and M12) [D2.1.1, D2.1.2]. In this section we summarize requirements for Commius software architecture.

Requirements are obtained by analyzing initial stories, provided by the consortium partners and a pool of external users (mainly from Greece and Spain), for the description of the most important processes in order to:

- Acquire detailed information about the steps needed for the completion of the processes.
- Reveal the most important interoperability issues that Commius will have to address.

The analysis resulted into the definition of the Commius workflow, i.e. the main phases which represent all possible groups of activities that the users perform using the system. These phases, as illustrated in Figure 3, were the input for the definition of the use cases and furthermore the functional and non functional requirements of the Commius system.

![Figure 3: The Commius general workflow](image)

© Commius Consortium
• **Setup and parameterization phase:** Refers to all activities for the setup of the Commius system and the storage of the template models that will be used by all collaborative partners. Under this phase, the functionality provided by Commius will have to address the following:
  o Management of the Commius content (i.e. profiles, document templates, process template).
  o Management of the Commius Partners and Users.
  o Authorization & authentication of Partners/ Users (through PKI infrastructure, etc.).
  o Management of external systems (including legacy systems)
  o Management of networks of Commius-based systems

• **Monitor collaboration:** Refers to all necessary activities for the collaboration establishment among two or more Partners such as:
  o Search for new partner(s) for business collaboration.
  o Mapping of the necessary information between the different enterprises which will interoperate.
  o Support for secured and trusted collaborations.
  o Support for interaction with external systems.

• **Workflow execution and monitoring:** Refers to the core actions done by the Partners, such as:
  o Prepare and send a request for a business transaction.
  o Receive and process a request according with the agreed business process.
  o Collaborate to the production of artifacts.

The requirements analysis has revealed the need for an open architecture, where Commius will have to be installed locally and be supported by external distributed services, which might be handled by external providers. For instance, these services can provide the following supporting functionalities:

• Provision of the templates, and various parameterization information (registry of Partners, document and process templates).
• User authentication and authorization.
Figure 4: Architectural requirements based on the Commius Workflow

The need for a centralized support is more in the initial phase of the Commius workflow, where a partner will have to register itself and get all necessary information. As we move towards the collaboration establishment local Commius systems will have to collaborate together for the exchange of necessary information; in parallel the need for the centralized functionality remains, as the partner will have to search through the registry of any other candidate collaborative partner. Last for the workflow execution, this is being done among the local Commius without the need for a centralized support (e.g. main exchange, etc.). The need for a centralized mechanism is only in case of inconsistencies among the template models used.

From the requirements, the Commius architecture will have to provide both P2P (peer-to-peer) and centralized models, according to the needs shown in Figure 4. Thus, the design of Commius must be flexible so that Commius can be configured for different models. In this deliverable, the conceptual architecture of Commius will address the requirements for setup and parameterization phase, monitor collaboration, and workflow monitoring and execution by providing an open architecture for centralized and P2P models.
7 Commius Architectural Overview

7.1 Overview

Figure 5 presents an overview of Commius software architecture. The architecture includes the following parts:

- **User tools**: include tools supporting the user to perform business tasks and to configure Commius. Commius user tools will rely on existing email and Web browser and will not require any modification or plugins for the email client and Web browser. The user tools will be used to perform business tasks as well as to manage and configure Commius. User tools are discussed in detail Section 8.1.

- **Email Gateway Plugin**: is responsible for intercepting and post-processing emails. Email Gateway Plugin is discussed in Section 8.2.

- **Commius Modules and Module Management**: include system- and business-related modules that handle interoperability tasks represented by email messages passed through the Commius. Modules in Commius will be built from components which can be provided by different layers. Modules and Module Management are discussed in Sections 8.3 and 8.4.

- **System Interoperability components**: include components providing functionalities for achieving system interoperability. System Interoperability components are discussed in Section 8.5.

- **Semantic Interoperability components**: include components providing functionalities for achieving semantic interoperability. Semantic Interoperability components are discussed in Section 8.6.
• **Process Interoperability components**: include components providing functionalities for supporting process interoperability. Process Interoperability components are discussed in Section 8.7.

• **Data Management components**: includes components providing data management facilities which are needed by Commius. This will be discussed in Section 8.8.

• **Security and Privacy components**: includes components providing security and privacy infrastructure for Commius. Security and privacy issues are discussed in detail in Section 9.

• **Commius hosting environment**: is the execution environment which hosts Commius components, modules and services.

Software modules will have well-defined interface. They can be exposed as software services, thus allow them being accessed through the Web. Components interact with each other via component interface which defines data representations and operations to control the behaviour of a component. In the Commius architecture, we will not define and manage common data repositories. Instead, data will be exchanged via component interface. For example, common profile information of users can be managed by data management components.

Core entities of the Commius architecture can interact with external systems, which are not part of the Commius. External systems include common legacy systems in SMEs as well as specific SME legacy systems and other services supporting the business of SMEs. While the main focus of Commius is to support legacy systems, Commius will also provide means to facilitate the interaction between core Commius elements and other external systems, e.g. Web services-based document repository, as well. External systems are elaborated in Section 10.

The architectural overview above describes a single Commius system which can be deployed for a single SME. Such a deployment can be done by the SME itself or by a Commius Service Provider. Individual Commius systems can be connected to establish a cooperative network of Commius to support a network of SMEs. We discuss the network of cooperative SMEs in Section 11.2.
7.2 Overview of Emails Handling Process

Figure 6: Overview of how Commius handles emails

Figure 6 describes how Commius handles emails. When an email arrives, the Email Gateway Plugin will extract metadata (context data) from the email and its attachments, such as sender and receiver ids, existing Commius embedded header information, and relevant keywords. The extraction is performed by applying regular expression rules which are predefined and updated regularly. The extracted metadata, a set of keys and values, is used to understand the type of document represented by the mail (i.e. an order, a request for quotation, etc.). Exploiting the type of document represented by the mail and other available metadata in the mail is considered in the context of a particular SME business process. Alongside the processing of the email, Commius will collect additional, context-sensitive information, by exploiting the process layer and the corresponding business module itself. In Commius, the selection of business modules can be automatic based on specific rules, and/or metadata enriched with other metadata related to business activities or be interactive by asking the user to select appropriate modules based on Commius recommendations.

The email and additional information resulted from the processing of modules are forwarded to the Email Gateway Plugin which combines and prepares the result in a right format before sending the resulting email(s) to appropriate recipients.

7.3 Example of Commius Usage: an Illustrative Partner Search Scenario

In this section, we describe an illustrative scenario discussing how Commius can be used from software point of view. This scenario is useful to understand the overall behavior of the Commius system\textsuperscript{17}.

\textsuperscript{17} Another detailed scenario describing the interaction among software components will be given in Section 8.10.
Figure 7 presents the illustrating partner search request, which includes many steps. The Sender opens the mail client and prepares a mail containing some criteria to find a new partner to provide a particular service, which is then forwarded to the Sender’s mail server which is ‘COMMIUS enabled’. An Authentication/Authorization module identifies that the Sender is an authorized COMMIUS end-user and the mail is recognized as a Partner Search request and handled as a step of the “Looking for new service providers” SME business process. Therefore, the Partner Search business module is exploited to enrich the mail and to build a list of contacts. To provide feedback to the Sender (as a reply message to the message that has been sent by the sender), the following actions are being executed. In the Information Extraction phase Commius determines that “criteria based contact retrieval” must be accomplished, thus it exploits a Contact Retrieval business module. Fortunately, such a module is installed and registered. The Contact Retrieval module utilizes a CRM system connector (one of System Interoperability components) which retrieves contacts that Sender’s company is related to. The Email Gateway Plugin forwards an automated email reply to the sender with the existing contacts that assumed to be relevant. The Sender can review the email and accept or reject the automated email reply. This scenario illustrates the use of individual Commius.

Consider the case when the automated email reply is not satisfied the Sender. The Commius of the Sender can look for the network of cooperative Commius by sending the request to another Receiver. When the email arrives at the mail server of the Receiver, the mail server of the Receiver is also Commius-enabled, the Authentication/Authorization module at the Receiver’s side examines the Sender’s private key and verifies that Sender’s mail is not a hoax mail so it must be subjected to processing before it arrives at the Receiver. The Receiver is being informed about the new request via a confirmation email. The Receiver decides to accept or reject the email request by replying the confirmation email or clicking the embedded link in the confirmation email using the Web browser. In case of rejection, the Receiver’s Commius sends an automatic response to the Sender about the rejection of the request. In case of acceptance, the Commius at the Receiver does similar steps as in the case of individual Commius. Before sending the result back to the Sender, the Email Gateway Plugin forwards an automated email reply to the Receiver with the existing contacts that assumed to be relevant. The Receiver then can either send the proposal of contacts to the Sender and/or enrich the proposal with any further (manual) input. The Receiver can also reject the result and send the feedback email to the Sender. This scenario illustrates the case of using a network of cooperative SMEs enabled with Commius.
Figure 7: Search partner scenario
8 Detailed Design of The Commius Architecture

8.1 User Tools

The user will use tools to utilize Commius features as well as to manage the Commius system. Typically the user will use Email clients and Web browsers to utilize Commius utilities. Furthermore, the user and the administrator of Commius will be allowed to perform some advanced and administrative tasks that require some Commius-specific tools. Hence, we also distinguish between “novice users” who just use email and web browser to conduct normal business tasks and “advanced users” who can use complex tools provided by Commius, such as ontology mapping tools.

There would be some tasks where different types of user tools are involved. For example, a user can receive an email in which there is an embedded link to a resource identified by an URI and a browser can be invoked to access the resource via the URI. Therefore, interactions between email client and Web browser will be supported.

The user will use mail clients and web browsers as normal to use features of Commius. There will be various web-based tools for the user to manage and configure the Commius system as well as to perform some activities in his/her businesses, such as browsing detailed information given in an embedded link. There will be no change in mail clients or Web browsers. The Commius will develop web-based tools for various tasks. Most tools will be Web-based, thus the user can easily use them.

8.1.1 Tools for assisting end-users to perform their business

The major tools for assisting the end-users to perform their business will be built atop business modules (see Section 8.3.2) which will be developed later. Therefore, at this time of writing, we could not list all potential tools for the end-users that will be examined in detail later on. Currently, we envisage the following Web-based tools for assisting the end-user to perform their business.

- **Task Recommendation Tool**: this tool will recommend relevant human tasks in a process and provide extra information for a task.

- **Task Tracking Tool**: this tool will allow a user to track performed steps in a running process instance or in finished processes. This feature will give the user the opportunity to directly embed user history or useful process information into a Commius enhanced email. For example the system could display the current stage of an order or past customer contacts including their content could be listed to improve support processes.

These user tools will be built atop business modules and expose features provided by business modules.

8.1.2 Tools for managing and customizing Commius

Currently, we foresee the following tools for managing and configuring the Commius system.

- **Tool for pattern creation**: This tool will be used by customization specialist, system administrator or advanced user to define or customize regular expression, XPath or
other patterns. Such patterns will be used for detection of business objects in text of email or attached files.

- **Concept matching tool**: This tool will be used by business users with some experience of using the system and some understanding of document structures (information and fields contained in a document). The tool will use a graphical interface to facilitate the mapping of an incoming document or a document type from external document ontology to a document type from the internal document ontology of the company (see Section 8.6). At configuration stage, the tool should also allow a company to create its own document type ontology using information from legacy databases or other legacy sources, and building upon a core ontology of business documents provided by the Commius interoperability utility.

- **Tools for managing modules**: There will be a GUI interface to manage the installed modules (adding, removing, eventually starting and stopping).

- **User Management Tool**: this tool is used to manage user and partner profiles and access controls. By using this tool, we can easily manage access controls in Commius from remote site. This tool will be developed based on security components.

- **The Process Configuration Tool**: this tool allows the user to comprehend his Company-specific business processes comfortably. The tool will provide standard process templates like for example selling, invoicing or support issues. Using either a drag and drop functionality which displays the process as a visual process chain or a questioning system, which guides through the configuration process, the user will be able to intuitively adapt his/her specific processes based on the templates. The adapted process will be stored in an enterprise process repository which is being introduced in Section 8.7.

These tools will be used mostly by advanced-users, system administrators and integrators, and service providers to ensure that the Commius is property configured and operational.

### 8.2 Email Gateway Plugins

Email Gateway Plugins are used to intercept email, extract relevant information and annotate emails with new information. In Commius, we will implement some prototypes of Email Gateway Plugins for contemporary email servers such as Postfix and Microsoft Exchange servers. Emails received by a mail server will be forwarded by an SMTP Injector into the core component of the Email Gateway Plugin. After being processed by Commius modules, the emails will be reinjected back to the mail server for final delivery. Figure 8 describes this process.

![Figure 8: Design of a Mail Gateway Plugin for Postfix MTA](image-url)
The Email Gateway Plugin in Figure 8 is implemented as a generic content filter. All interfaces in this setting are built above the SMTP protocol. Commius Mail Gateway Plugins will have two interfaces, one to the real mail server and the other one to Commius modules. The two main functions of a Commius Mail Gateway Plugin are:

- To intercept emails and pass Commius-related emails to corresponding modules.
- To relay emails given by modules to the email server.

8.2.1 Message Post Processing component

The message post processing (MPP) component is responsible for aggregation of XML messages produced by email processing modules into suitable modified email. MPP component will also implement dynamic web based user interface (using AJAX\textsuperscript{18} technology) that allows more sophisticated integration of module results. This user interface will be accessible via URL links embedded in the modified email message. The components of MPP are depicted in Figure 10.

\textsuperscript{18} http://en.wikipedia.org/wiki/AJAX
• Message Processor – application of specific post-processing features upon module results. Possible features cover message filtering, list sorting, layout and style settings;
• Session Manager – temporary storage of modules’ results processed from the same email. The data from the same email are aggregated via SessionID values generated uniquely for the email;
• Email Processor – aggregation of modules’ results into plain-text message used by email server. The messages can be enriched by links to web interface providing more a convenient way for viewing module results.
• UI Servlet – aggregation of modules’ results into HTTP output. This sub-component is responsible to layout the specific module(s) in dynamic HTML page. Each module must reference client side application for convenient output data processing (in JavaScript). The component is responsible for data feeds (built as servlets, GWT\(^\text{19}\) services, etc.) to provide client application with up-to-date data;
• Configuration – sub-component allowing users to customize output messages and module settings.

## 8.3 Commius Modules

Requests coming to Commius will be handled by Commius modules. We categorize two types of modules: system-oriented modules and business-oriented modules. System-oriented modules are used to perform tasks related to Commius operations, for example, module and component discovery, and authentication/authorization support. Business-oriented modules implement business functionalities offered by Commius. Requests coming to the Commius system will be handled by modules which compose and execute various, different components. Business-oriented modules will be developed to support different business purposes.

### 8.3.1 System-oriented modules

#### 8.3.1.1 Authentication/Authorization

This module is used to perform authentication and authorization. A role based access will be created for Commius. This access will be administrated by this module.

The mandatory profile that will be defined consists of four roles: administrator, super-user profile, normal user and final user. This is only a basic and initial creation of this profile.

- Commius Administrator: will configure Commius modules and all task related with the configuration of the server (batch, backups, database administration, processes repository administration, etc.)
- Commius Superuser: will give access to the different modules to different user and manage the access to the server and the different modules of the server. The task of the Commius Superuser is not related with the management of the different process information, it is only the management of the different users that can access to Commius server and use the Commius application.
- Commius user: will receive an e-mail from the client user and will start Commius to support the client user.

\(^{19}\) [http://code.google.com/webtoolkit/](http://code.google.com/webtoolkit/)
Client user: is the SME that sends the initial e-mail. It will receive the enhanced e-mail from Commius.

The relationship between the Commius user, the client user and the process will be gathered in the repository (or the information of this process). Commius will assign a user that can start each process stored in the repository of processes so that those users will be able to manage the classified information of the process.

A management process to create, delete or modify users inside the system must be defined. The module of Authentication/Authorization will be also exposed as a service so that it can be accessed remotely.

8.3.2 Business modules

Commius follows the paradigm of almost zero cost of entry. Therefore, we do not only develop the technical infrastructure that is needed to implement interoperability. Rather, Commius also provides ready-to-use, business-oriented packages. Thereby, potential users have the possibility to install an almost fully preconfigured Commius system that can enable interoperability instantly. The packages themselves consist of business modules and system modules. Business modules provide the facilities and functionalities for the domain and business-area where Commius should be used. System modules contain interfacing software components for the system environment, i.e. the IT systems that should interwork with Commius.

Business modules are preconfigured packages that fulfill specific tasks in the context of a defined set of business processes within a determined business domain. These modules consist of a set of software components. They are needed to provide the business support as well as of a set of configuration data that sets up the components in order to operate accordingly to the process suitable for domain needs. More particular, business modules can be decomposed into:

- Software components (email plug-in, user management, process interoperability component, etc.)
- Business domain adapted semantic configuration data (domain ontologies, specific business document ontologies, etc.)
- Business process configuration data (reference business processes, process patterns, common adaptation schemas, etc.)

An example for a business module is “procurement of materials”. Such a module provides a set of functionalities to support users in the business process of disposition and operations of buying needed physical products. However, as the module is process specific, it does not provide support for the procurement of services, since the process of buying this special kind of product differs from the materials-buying-process.

Modules may also be exposed as Web services, thus allowing them to be accessed through the Web. This allows us to support different types of interactions with the user of Commius. There are many tasks the user will conduct via emails, which a module can handle by processing emails, and there are tasks where the user can simply interact with the module through a Web browser, for example, to confirm a step while the module handling the request.
The Commius module model supports the end user, the developer, and the service provider to use and build their solutions. For example, a provider can implement a module and specify module description and business process templates guiding the operation of the module; or the developer can simply use an existing module but provide new business process templates. The end user may just need to obtain modules and change module configuration and business process templates suitable for his/her environments. Thus, business objectives are achieved through the implementation of modules and/or the configuration of business process templates.

8.4 Commius Module Management

The Commius Module Manager is responsible for managing modules available in Commius. Its tasks include adding, configuring and removing modules, discovering and instantiating modules.

8.4.1 Commius Module Manager

This module is used to manage other modules. All modules in Commius will register themselves to the Commius Module Manager. Based on the configuration, the Module Manager will invoke corresponding modules to process requests sent to Commius. To be managed by the Module Manager, a module must follow a pre-defined interface. The Module Manager will also be exposed as a service so that it can be configured from remote site.

Figure 11 describes information repositories used by the Module Manager.

- Rule Repository: includes possible pre-configured rules for selecting modules.
- Module Repository: includes modules
- Module Business Template Repository: includes business templates used by modules.

![Figure 11: Module Manager and dependent repositories](image)

The information in these repositories will be updated by using corresponding tools. For example, a module and its relevant information can be packaged, uploaded into the Module Repository and executed on demand.

Based on the enriched metadata, the Module Manager performs a matching process to select the right module to handle incoming email. This matching process utilizes various sources of information, including extracted keywords, module description, historical data, rules, and components, and in particular, semantic information.

- Selecting a module based on specific rules: for example, based on pre-defined rules and metadata a module can be selected. This happens when the user knows for sure which modules should process which types of emails.
- Selecting a module based on enhanced metadata: the metadata from email extraction might not be enough. In this case, the Module Manager can enhance the metadata by
invoking relevant components and modules. Then, module selection can be based on the enhanced data.

- Selecting a module based on user decision: when selecting a module, the Module Manager can also interact with the user, in case it cannot decide the right module (e.g., due to rule conflict or missing information). In this case, the Module Manager will inform the user with an email including embedded links and by replying the email or clicking the links, a module can be selected.

### 8.4.2 COMMIUS Module Discovery

This module is used to discover other modules and components available in the Commius system. It works closely with the Module Manager. The discovery will be conducted based on metadata specified in the deployment of modules. When a module/component registered into the Commius system, it is required that the module/component provides enough metadata to support the discovery. Metadata will be defined based on a specification which will be developed in WP4. The Module Discovery will support different means to find relevant modules and components by providing notification and query mechanism.

### 8.5 System Interoperability

The system interoperability layer in Commius will provide a basic interoperability infrastructure over SMTP to:

- Extract information from emails and annotate emails with new information
  - Extracted information will be passed to Semantic Interoperability Layer
  - Annotated information will be obtained from Semantic Interoperability Layer
- Integrate with external systems in order to access legacy/external information and services

The main components of System Interoperability layer includes

- Message Decomposition (attachments, body, header, threads),
- Information Extraction, and
- System Connectors

These components will be used as building blocks for interoperability modules which will implement concrete system interoperability functionality.

### 8.5.1 System Interoperability Components

![Figure 12: Component view of system interoperability](image-url)
8.5.1.1 Message Decomposition

Each email passed through Commius will be analyzed, archived and decomposed. Decomposition means decoding mime email messages to separated parts such as headers, email body, email attachments of various types and its text versions. Thus, other Commius components can directly request needed attachment, email header or text of the message without need to understand email MIME and other standards.

8.5.1.2 Information Extraction

Information extraction components will use, adapt and deliver new methods based on state of the art in information extraction and information retrieval fields. We will adapt such methods on email content based on analysis of email communication on Commius pilots and available business email communication such as Enron corpus or W3C email corpus available from TREC conference\(^\text{20}\) and its Enterprise Track\(^\text{21}\).

Based on our analysis we will start with pattern based information extraction using regular expressions and XPath (for information extraction from XML and HTML attachments e.g. ebXML). This will allow us to deliver results by analysis of system-to-human email communication such as notifications, invoices, payments, orders and other communication where part of it is well structured and information can be extracted using predefined or learned patterns. The patterns can be defined by using tools for pattern definition, e.g. by selecting similar texts. Patterns will be aligned with ontology or organizational model – thus extracted data will be related to concrete business objects. Information extraction components will be connected with Semantic Annotation in the Semantic Interoperability Layer.

Input for information extraction components will be plain text from email message or attachments converted to the plain text, HTML or XML. Output of information extraction components will be key - value pairs which will represent business objects or its properties. Key-value pairs can be further transformed using specialized transformers to connect data using System Connectors.

The information extraction components will be built on previous work of IISAS – Ontea [L+07] which is published as open source\(^\text{22}\) - and we will use Ontea for pattern and transformers interfaces.

8.5.1.3 System Connectors

System connectors are used to interface to external systems, including legacy software, required to be accessible by Commius modules. The most common external systems required in Commius are given in Section 10. System Connector components must provide Commius with the ability to access selected functionality of external systems. External systems are invoked from Commius components based on application logic, defined in the components.

Challenges posed by System Connectors task are following:

\(^{20}\) http://trec.nist.gov/
\(^{21}\) http://www.ins.cwi.nl/projects/trec-ent/wiki/index.php/Main_Page
\(^{22}\) http://ontea.sourceforge.net/
• Common description of connector capabilities – connector capabilities must be described in uniform way to define available functions/input/output data and provide common way of utilization in the Commius modules.

• Extensibility – due to the heterogeneity of legacy systems, it is hard to provide 'universal' connectors for legacy application classes; the effort will be to provide connectors able to adapt to concrete systems by connector configuration – to minimize the need to modify/rewrite the connectors source code.

• Security – connectors must be able to exploit legacy systems, where the security tokens are mandatory to perform required operations. The security concerns and the proposed approach to the security in System Connectors are described in Section 9.

Figure 13 depicts the component view of the System Connector. System Connector component contains connection logic and configuration. The System Connector component will use appropriate External System Client to interact with corresponding External System. The System Connector logic is responsible for preprocessing of input data, retrieval of data/information from legacy systems and post-processing of extracted data to the required form. The configuration defines parameters required by External System Client as well as System Connector configuration (e.g. RDMS System Connector configuration may contain RDMS client configuration (server, port, and database) and definition of a query to be issued from the System Connector). The External System Client provides mechanisms for connecting and retrieving data from an external system.

Figure 13: Component diagram of System Connector
8.5.2 System Interoperability Control and Data Flows

Figure 14 shows the operation of the system interoperability layer. The Email Gateway Plugin is triggered when an email comes to the SMTP server and then the email is processed by other system interoperability components as needed.

Email is received by SMTP server and decomposed by the Email Gateway Plugin which stores every part of the email including the header part. Based on the extension of the stored parts of the email several convertors will be started. Converted parts (attachments) will be processed by identified modules which will be started in hosting environment. Outputs of these modules will be gathered by Email Gateway Plugin (also with Message Post Processing component) and included as a link or inline text in email message. This email enriched by the Commius system will be delivered back to the SMTP server.

8.6 Semantic Interoperability

This layer includes components to
- achieve semantic alignment;
- facilitate concept negotiation; and to
- annotate messages with meta-data to embed semantics within them.

From an architectural point of view, the development and implementation of this layer follows the general Commius approach.
Initially in the semantic interoperability layer, documents are annotated and mapped. The UN/CEFACT "Core Components" standard functions as an interlingual for this mapping process. Thus in the first instance, “concepts” relate to documents: specifically the tokens of information exchanged through a given document as a communication medium. If a document from a sending SME cannot be mapped into a document type of a receiving SME, then the content of the document must be negotiated.

This begins with the sending SME identifying the essential information in the original document: that is, the minimal information that must form part of any replacement document. “Superfluous” information is removed and the resulting set of information tokens considered: this is viewed as a “super-concept” of the original document. If this can be mapped into a document type of the receiving SME, then this alternative is used as a replacement in the communication. If it is not possible to render this into a document type, then the sender and receiver negotiate over which structure to use: in the simplest case, the receiver proposes a “super-concept” of the original document and the sender identifies the additional information needed to enrich this structure so that it includes the essential information. Once this substitute structure is established, then the receiver will consider the likelihood of his receiving future communications from the sender: if it is likely, then it will enrich its document type repository with the substitute; otherwise, the substitute will be used only temporarily.

8.6.1 Semantic Interoperability Components

Figure 15 illustrates the components conforming the Semantic Interoperability Layer. The individual components are discussed below.

8.6.1.1 Semantic Manager

This component functions as the interface to the Semantic Interoperability Layer. Its basic functionality consists of managing all the semantic interoperability, from receiving a document for semantic parsing or annotation to initiating any needed concept negotiation. It provides the layer output as well consisting of the rendering of incoming document to the structure of document types specific of the Commius system.

8.6.1.2 Semantic Annotator

This component implements the annotation functionality of the SIL. The aim of the component is to assign a set of semantic tags to an incoming document in a way which would reflect the actual meaning of the document. This assignment provides a complete and consistent information about the content of the message. Additionally a set of tags are grouped into an ontology thus forming a hierarchy of concepts and relations used for semantically describing emails. Internally various methods of semantic annotation are used, which exploit state-of-the-art advances in pattern-based annotation and information extraction. The output of this component is called annotated document.
8.6.1.3 Document Interpreter

This component reads the structure of Core Components found in an annotated document and tries to match them to an appropriate document type specific for the company using Commius. For performing the matching, the Document Type Manager helps by providing an interface to the both the Semantic Core and the Document Type Repository. The output of this component is called mapped document and essentially represents the output of the whole Semantic Interoperability Layer.

8.6.1.4 Semantic Negotiator

On occasion, direct mappings may not be possible and an alignment of document structures will need to be negotiated. This component is in charge of reaching common agreements with any other Semantic Negotiator in another Commius system. It starts operating when the Document Interpreter finds it difficult to align the annotated document to a Document Type. In such a case, a common understanding is sought by finding a common set of Core Components to represent the same information.

8.6.1.5 Document Type Manager

This component works as an interface to manipulate both the Semantic Core and the Document Type Repository. It provides operations for the creation, addition, updating, and removal of Document Types. In addition, it further enriches the Semantic Core by allowing the addition of new documents, which help in the document classification for mapping purposes.
8.6.1.6 Document Type Repository

More than a software component, this is a collection of document types available to a Commius system. Document types can be seen as the categories in which a document can be classified, cf. invoice document type, quotation document type, etc. Furthermore, the document types function as document templates or a specific form in company which contains the “necessary” information for the user understanding.

8.6.1.7 Semantic Core

For the purposes of manipulating and comparing annotated documents, it is useful to make use of some alternative structure such as an algebraic structure called a lattice created by means of a Formal Concept Analysis mechanism. This lattice represents the Core Component relationships existing among documents previously mapped (document instances). See D5.1 for further information regarding the semantic core.

8.6.2 Semantic Interoperability Control and Data Flows

![Semantic Interoperability Control and Data Flows Diagram]

Figure 16 shows the operation of the Semantic Interoperability Layer within Commius. The layer is triggered when an incoming document (e.g. an email) has to be “understood” or classified in the company's terms and structure. The purpose of this is to support of the Process Interoperability Layer in determining a specific process based on a document type.
After the arrival of an incoming document, the Semantic Annotator extracts information tokens based on the UN-CEFACT Core Components. Once these have been identified, the Document Interpreter attempts to locate an appropriate local document type with the aid of the Document Type Manager. If a document type containing these tokens is found, the information is mapped into a copy of the document type, i.e. a document template, and is returned as the layer output.

If no appropriate single document type can be found then the Semantic Manager investigates whether the email sender has Commiux installed as well (this could done by adding a special tag to outgoing emails). If the sender party has no Commiux then a list of potential document types is returned as the layer output. These document types could be used by the Process Interoperability Layer to determine potential processes.

On the other hand, if the sender party has Commiux installed then an automatic email-based negotiation process takes place between the two Commiux systems. There are three possible end results of this negotiation:

- The document type is “explained” to the receiver party who adds it to the Document Type Repository. Then the information is rendered into this new document type and presented as the layer output. This case happens when the new document type is considered of
particular importance by the receiver, perhaps because they expect to frequently receive it in the future.

- The negotiation reveals that the sender could use an alternative format which the sender can map to one of its document types. In this case, the document is re-sent using the alternative format.
- No alternative format is located and the receiver does not understand the document structure. In this case, a list of potential document types is presented as the layer output.

In this way the set of document types recognised by the Semantic Interoperability Layer is allowed to naturally grow.

Further information about the Semantic Core can be found in D5.1 (internal version) due in M12. Specific details of all the other Semantic Interoperability Layer components will be found in D5.2 due in M22 (initial version).

8.7 Process Interoperability

The Commius Process Interoperability will be supported in two phases, build-time and runtime. The support in each phase will be realized by a software module which consists of several components. In the build-time phase, a process configuration module is used to adapt general processes until they match the specifications of an individual company. A configuration tool will be provided to the user to support the configuration. In the runtime phase, another module will support business processes.

In order to immediately assist the user, the Commius processes have to be highly adaptable in a fast and user-friendly way. Hence a process configuration tool has to be developed which enables the user to comprehend his/her in-house processes comfortably. Therefore, Commius users access a reference model directory which provides adaptable templates for different types of standard processes like selling, invoicing, etc. Based on the provided standard templates, the user customizes the reference models manually until equalization with its own processes has been achieved and the customized process has been stored in an enterprise process repository. The customizing can either be realized through a questioning system or an easy-to-use graphical modelling interface, based on drag and drop. This customizing Toolset is also applicable to create specialized processes with no matching template. Figure 17 depicts the configuration process.
After the Commius Process Database has been customized, Commius can immediately support the in-house processes. This supporting functionality can be structured into different components named: Enterprise Process Repository, Process Instance Database, Detection, Tracking, Assisting, and Advising.

### 8.7.1 Process Interoperability Components

The four components of the process interoperability layer are also shown graphically in Figure 18.

#### 8.7.1.1 Enterprise Process Repository

This component is used to store process types available in Commius.

#### 8.7.1.2 Process Instance Database

This component is used to store process instances of process types available in Commius.
8.7.1.3 Detecting

In this task the Commius system checks whether the incoming semantically enhanced email can be allocated to an existing process instance from the Enterprise Process Repository. At this point, the System Interoperability layer has already decomposed and the Semantic Interoperability layer has semantically enhanced the email. There are two possibilities the incoming document can be processed. In case the Semantic Interoperability layer has already tagged the incoming email with a corresponding process ID and then the email can immediately be assigned to its overall process instance. Otherwise, the detection operation has to assign the incoming email to an appropriate process type based on the information detected by the Semantic Interoperability layer.

If no suitable process instance in the Enterprise Process Repository can be located, a new process instance is being created based on the information provided by the Semantic Interoperability layer and the predefined process type templates. Other incoming emails concerning this particular process will be allocated to this initial process instance henceforth.

8.7.1.4 Tracking

The process tracking functionality is responsible for monitoring all steps occurring within a process instance. Every exception concerning a single process instance will be assigned respective to its corresponding process instance. Following this approach, every performed step within a process is documented and comprehensible for further analysis.

8.7.1.5 Assisting

The process assisting functionality consists of two segments. One provides case related information about the particular instance, like customer history, contact information or useful web links while the other procures relevant process history from Process Instance Database. The decision which information is being displayed depends on the evaluated data from the Semantic Interoperability layer and the customized process type templates stored in the Enterprise Process Repository.

8.7.1.6 Advising

The process advising functionality prepares recommendations for further steps in a particular process instance. Since Commius already identified the correct process type, it can revert to knowledge about the identified task from the process database to provide suggestions about further steps. If the incoming email has, for example, been identified as a confirmation of payment of an order transaction, Commius would recommend triggering the shipment and procure reasonable options like forwarding the confirmation to the shipping department.

After passing these four kinds of operations, the document is being enriched with auxiliary information and forwarded to the System Interoperability layer which processes the email in such way that the respective user can access the ascertained information.

8.7.2 Process Interoperability Control and Data Flows

Figure 19 shows the operation of the Process Interoperability layer within Commius. As soon, as a semantically enriched document reaches the Interoperability Layer, the incoming process type is being detected. As already stated in Section 8.7.1.1, there are two possibilities to handle the incoming document: either the incoming document concerns a newly-created process instance or an initiated process instance.
Subsequently corresponding information depending on the currently performed step within the process instance will be collected and attached to the document. Then, matching advices from the process database concerning the current process will be identified and included into the document. Next, the collected additional information as well as the advising information will be handed in a XML document to the System Interoperability layer, which processes the document into the final enriched email.

Figure 19: Process Interoperability control flow
8.7.3 The Interorganizational flow

This section will present how, with the use of Commius, cross-organizational process interoperability could be realized. In order to exemplify the appliance possibilities of Commius, we will use an illustrating example to show the utilization of each Commius components as well as possibilities to use Commius to support an interorganizational business process.

In a scenario where two or more Commius users are involved within a business some internal process information of an SME have to be transparent for third user parties or business partners and have to be exchanged. This kind of data exchange will mainly take place within the tracking, assisting and advising operations. Therefore, Commius will provide a standardized gateway to exchange data within two or more Commius applications. Figure 20 illustrates a use-case-scenario in which a complete ordering process is being supported by a Commius system. The incoming email order of the manufacturing user is being intercepted by Commius. Supported by the Semantic and System Interoperability components, the email is being transformed into a standardized XML file with semantically enhanced tags and identified a new process instance with the process type "Order". Commius checks whether the ordered items are on stock and generates a fully automated reply with relevant information, e.g. order confirmation, preliminary delivery date, and payment instructions. Additionally, Commius enriches the created XML file with additional information needed by the processing department like customer history or status of payment. Subsequent to the reception of the payment the ordered items are ready for shipping and the Commius forwards the required information, like order details and delivery address, to the shipping department in which the shipping process can be triggered.

Even only one collaboration partner makes use of Commius, for example, the Manufacturer in Figure 20 is not using Commius, the Manufacturer will still be able to receive the order confirmation as well as possible extra information. Since Commius uses email as the main communication medium, information concerning the collaboration process can be exchanged within partners even if only one partner makes use of Commius.
8.8 Data Management Components

There are many types of information shared among components and modules. We identified the following information:

- Email objects and metadata: the email object represents the email content and its attachments. This type of objects is processed by and transferred between main components, such as MessageDecomposition, SemanticAnnotator, and ProcessHandler. During the processing, this type of objects might be stored as well. Various types of metadata exist. They will be represented and managed together with email objects.

- Business process templates: business process templates are associated with modules but are customized for different clients and businesses. There will be various components and tools accessing and modifying business process templates.

To simplify the data management of Common data, Commius will provide lightweight data repositories for storing and managing common data required. To facilitate the data exchange, common data will be described in XML. Therefore, data repositories will support XML data serialization.
8.9 Hosting Environment

The hosting environment is used to host software modules and components of Commius. The hosting environment is dependent very much on the technologies being used. The main functionality of the hosting environment is to serve as a container on top of which Commius modules and components are deployed and executed. Details of possible platforms and technologies for the hosting environment are described in Sections 11 and 12.

8.10 Example of a Business Module

To illustrate the dependency among components and modules, in this section we present an illustrative example of Business Modules.

We consider an email specifying an incoming purchase order which is as part of an incoming order management process. We suppose that for a SME the management of an order is split in three phases: Incoming Order Management, Payment Reception, and Shipment. For this scenario, we illustrate the first phase, the “Incoming Order” phase, in which the SME receives the order and start to process it. The Incoming Order Phase is implemented by a business module named “Order Receiving and Enriching”. The purpose of this module is to add some information to an incoming order to support the SME order management process. In particular the module adds information on the customer sending the order, on the product requested, etc.

Figure 21 describes the “Order Receiving and Enriching” module, how it depends on other components and how Commius selects and executes this module. We explain the sequence of execution in the following:

Step 1: Producing metadata by Message Decomposition

- The email is passed to the Message Decomposition which does
  - invoke InformationExtraction with the email
  - the InformationExtraction returns metadata (from, to, subject, much_more) and attachments

Step 2: Find and execute relevant module

- The Module Manager obtains the mapped document (mails, metadata, etc.) and performs the module selection.
- The “Order Receiving and Enriching Module” is found and executed by the Module Manager

Step 3: The execution of the “Order Receiving and Enriching Module”

- Invoke the SemanticManager component with Metadata+attachments+emails
  - The SemanticManager component invokes the SemanticAnnotator which returns the original email as an annotation, such as Keyword Product ID, Keyword Amount, Keyword Customer, Keyword Bank Account Number, Bank Identification Code.
When the SemanticAnnotator processes the input, it either looks into the configuration (or ask Module Manager) the information about Product database and Customer database and gets extra information via System Connectors or invokes the System Connector with some information and the System Connector will return the extra information.

- Then, The SemanticManager component invokes the DocumentInterpreter component by providing the original email as an annotation including **Keyword Product ID, Keyword Amount, Keyword Customer, Keyword Bank Account Number, Bank Identification Code.** The DocumentInterpreter returns a mapped document containing the following information: the Document Type description, i.e. purchase order, product ID, the amount, the customer name, bank account number, bank identification code, and possibly one or more process phase type IDs identifying the document as potentially belonging to one or many process phases.

- The Semantic Manager returns the mapped document
- Invoke the ProcessHandler component which will
  - Invoke the Detecting component with the original email as an annotation, **Type Order Process, Process ID, Keyword Product ID, Keyword Amount, Keyword Customer, Keyword Bank Account Number, Bank Identification Code**
    - The Detecting component determines if the process is already known and running. If it is a new process instance, a new process will be assigned to it.
  - Invoke the Tracking component with **Process ID, Product ID, Amount, Customer, Bank Account Data** which annotates the original email. The Tracking component determines the current stage of the process and tracks each performed process step.
  - Invoke the Assisting component with **Process ID, Product ID, Amount, Customer, Bank Account Data, State** which annotates the email
  - Invoke the Advising component with **Process ID, Product ID, Amount, Customer, Bank Account Data, State, Number on stock, customer status, recently ordered items** which adds a mail-to link to confirm the payment on this order
  - Now with the information returned by Advising component, the EPR email-template assigned to this type of process is used to create the output file, which contains all additional information, advice and the original email.

- The module returns all this information within a standardized XML schema to the module manager

### Step 4: The composition of results

- The Module Manager receives the returned information from the “Order Receiving and Enriching Module”
- The Module Manager invokes MPP with the received information from Step 3
- The MPP builds emails and sends emails to the customer
Figure 21: (Simplified) sequence of the Order Receiving and Enriching Module
9 Security and Privacy Concerns

9.1 Security concerns

As a general principle, Commius will try to provide a medium for SME’s to interoperare in a way as secure as possible, because Commius can be used to perform operations which might involve economic transactions hence it can be considered to be especially likely to suffer security attacks. We distinguish security and privacy concerns within Commius software components and between Commius software components and external systems (such as legacy systems). Within Commius, we provide mechanism for exchanging security and identity information between different components and modules. Between Commius and external systems, we identify the following points where security should be addressed:

- Mail client and web browsers to a Commius system.
- Commius system connectors to external systems.
- Identity management for Commius users.

In this section, we analyze the main types of attacks/threats that the Commius system could suffer and explain briefly the level of coverage for them and the measures adopted. Most types of attacks/threats are well known for any network and email-based systems but we will analyze them in the context of Commius.

**Man in the middle attacks:** an attacker manages (possibly by phishing techniques) to redirect communication through its own server and reflects any client/server messages to its original destiny, but listening them. If the conversation is not signed by a digital certificate, the attacker might change the contents of a message after the communication was established, allowing him, for example, to change a count number. This is possibly one of the most dangerous types of attacks, but it is not likely to occur inside the organization. To avoid it, we impose a constraint on Commius that all Commius communications must be signed with a digital certificate on a SSL protocol (for instance, https). Besides, such communications must be authenticated by both server and client. This will mitigate the risk that an attacker might imposture another Commius server in a distributed operation.

**Decrypting:** an attacker might be able to decrypt the content of a message, and then could access confidential information and even change it. Commius tries to prevent this event by combining SSL and digital certificates in all transmissions whenever possible, because SSL transmissions are encrypted. Besides, every store used by Commius must be encrypted to prevent it from being altered by an attacker.

**Deception:** an attacker might log into the system as a lesser privileged user and try to change his/her rights during the session. This threat will be addressed by using security protocols that will be defined after the network definition, because it depends on them.

**Disruption:** An attacker might cause disruption, for instance, by issuing a lot of service invocation to a server, consuming its resources and thus forcing it to fail. Denying service is one of the most common types of attacks, but not as harmful as the others, especially in a commercial environment. And the distributed nature of Commius might make it even harder...
to achieve. The initial approach of Commius will not cover this risk but be open for a later coverage, for instance, by adding black list filter to the server.

*Injection:* an attacker might modify results of a scripting page to include references to malicious URLs. A first barrier to these attacks is the encryption of the communication through https, but it might still be insufficient if the attacker gets to surpass it. Scripting vulnerabilities are one of the most difficult set of attacks to cover completely, because they are too many and new attacks arise every day. Because denial of scripting is probably not feasible in Commius, the last responsibility on this vulnerability would lie on the programming of these scripts, which should check any string injected into the page to avoid them to include any malicious URLs. As an intermediate approach, Commius might include some filters to allow only URLs to trusted sites. But it is still possible that more attacks appear. Commius prototype will not cover all these attacks, but will be opened to the inclusion of new filters to possibly cover them in the future.

*Pirate user defined tokens:* user defined tokens might change the level of security of the user logged, or include injected malicious URLs. This is very difficult to detect, thus it is firmly discouraged from the security point of view. It is better recommended to comply with a previously defined set of tokens, which must be defined by the Commius consortium. Including such tokens would involve the semantic layer to call the security layer to check the security of the token, and, if the token included some kind of URL, it would also imply some kind of very sophisticated algorithm.

*Modification of libraries:* An attacker might try to replace Commius programs/libraries by its own modified ones to let him, for instance, to redirect communication to a malicious server. This will be avoided by imposing the digital signature of all Commius libraries.

*Modification of configuration files:* An attacker might try to redirect communication to a malicious server by modifying configuration files. To avoid these, all configuration files will be stored on an encrypted store and accessed only through the Commius configuration tool that will sign it every time it changes.

*Modification of document templates:* An attacker might try to include harmful macros or changing the content of a document. To avoid this, all documents must be digitally signed by its originator, who will be the only person allowed to update them (that is, documents can be replaced only with other documents signed by the same entity/person)

*Inclusion of harmful macros in documents:* An attacker might try to include a harmful macro into a document generated by a document template (which might include legitimate macros). To avoid this, attached documents should be opened by the system disabling macros. That is, macros can only be included in documents templates and only for editing purposes. Besides, templates structure must mandatory include a password protection for protecting them from being altered.

*Commius email modifications:* This kind of attack is not strictly a responsibility of Commius. Each client is responsible for the security of their own machines, and that includes the use of an antivirus for preventing its mails to be altered. Hence Commius security will not cover such modifications of the emails.
Spam: it is the same for email modification. Avoiding spam is a responsibility of the system security, not Commius.

All other possible risks are not initially supported by the Commius prototype, but support for new risks might be added in future versions by the inclusion of new security filters as new risks prove to be worth to deal with.

9.2 Security and Privacy Solutions in Commius

Because the user needs to be authenticated and his/her access needs to be authorized, a public key infrastructure (PKI) might be required for a Commius-enabled system. And because Commius can communicate over a network with other Commius systems, a symmetric (double) authentication mechanism might be required. From the transport level, security will be based in the use of SSL for all communications when possible (that is, https for the plugins and SSL for connectors).

Commius will follow a generic security mechanism based on the inclusion of a chain of filters, which will provide support for further security mechanisms when the network architecture is defined. The chain of filters may also provide support for further transport security mechanisms. To add support for classified information, it is necessary that Commius security communicates in some way with the Semantic layer. To achieve this, the security layer will provide an interface for security classification upon token types.

Figure 22 illustrates the main figures subject to attack and the countermeasures adopted for protecting them. Pirate signs warn from likely hacker objectives. Note that process and repository are supposed to be located inside the SME network, so they are not initially considered to be protected directly (but they are indirectly protected by the signature of the Commius libraries). Also note that connections with legacy systems are especially dangerous because they might impose their own security, thus Commius might not have control over it.
9.2.1 Confidential Information Concerning Privacy Policies

One of the most important security issues Commius has to face will be the management of the confidential information inside the process management and the external systems. All the information that Commius extracts from the legacy systems must be classified and labeled. It is mandatory because Commius must know what kind of information it is managing (confidential, public, etc.) and if it is possible to show it to third parties. Without this classification it is not possible to manage the security issues (according to the Data Protection Law).

But this classification is impossible to be carried out without the message analysis which is performed by components in the Semantic layer. For this reason, and in order to separate the information security classification from the semantic layer, a security classification interface will be defined to allow the semantic layer to call the security layer to classify any field by providing its field type. Figure 23 illustrates the interaction between Security Layer and Semantic Layer and Figure 24 describes the corresponding sequence diagram for the interaction.
Anyway, one thing is identifying confidential information and another is to decide whether to grant access to it or not. This decision depends on the role of the user currently logged in the application and is closely tied to internal enterprise policies. Thus, Commius will not aim to make such a decision but will leave it up to the SME information system through a decision connector.

From the technical point of view, there are five possible key events when dealing with privacy data in the system, which are:

- Arrival of a mail to the system: whenever a mail arrives to the system, if it has been encrypted and signed with a Commius security component, the incoming message will have to be decrypted and its signature has to be confirmed prior to any other operations.
- Semantic analysis of the message: while extracting the semantic information from the incoming message, it is necessary to classify the security type of every token by invoking the Classify component of the security layer, and storing this classification as an additional semantic attribute.
• Storage of incoming data: due to current legislation about sensitive data, this sensitive data must be stored in a different way, depending on its type. For this reason, depending on types of data, incoming data will have to saved in a different store or even be saved in an encrypted way. Because this storage will normally be performed through the system connectors, which will be implemented as services, these services will be responsible for invoking the security components which provide encryption utilities.

• Sending output data: whenever the system sends new mails, these mails are subject to contain sensitive information. This sensitive information might be legally subject to encryption or even not allowed to be transmitted at all. For this reason, the MPP component should invoke the security filter to allow it to erase or encrypt existing sensitive information.

• Peer – to – peer data synchronization: should Commius systems decided to send internal data each other internally (without human intervention), this data might be subject to privacy constraints. For that reason, outgoing data should be analyzed and filtered in a similar way to the semantic analysis (see above). That is, all outgoing data should be classified and filtered prior to sending it to the peer.

• Distributed queries: whenever a system tries to query our Commius application about data, those data is subject to contain sensitive privacy data which should not be shared. Here the problem is similar to the “sending output data” event. All outgoing data should be categorized and filtered before sending it.

Figure 25 shows where the privacy concerns are localized and Figure 26 presents the steps performed by the system to deal with them.
9.2.2 Authentication of the user

One of the most important issues that Commius will face is the authentication of the request that the client user will send by emails to the system to obtain the information. Every user will be validated inside Commius through an authentication server (for instance a LDAP server). If there is not such a server installed in the network of the SME, Commius will install it as a part of the Commius installation procedure.

Commius might require that all Commius users have a digital signature associated and reject all requests / responses which are not signed with such certificate. This measure will allow a remote Commius server to be sure that a user is he who says that he is. Commius will also provide a wrapper for the main vendors of authentication servers (Active Directory, LDAP, etc.). The wrapper will include an API for accessing the attributes of the user and verifying that its digital signature is valid.
Commius needs the validation of every user because Commius manages confidential information of critical processes. Sometimes this confidential data cannot be sent to a non-certificated third party. Besides of this, because the information can travel through several Commius systems, it is also necessary and mandatory that the server identifies itself. This is achieved by the (optional in many servers) double identification mechanism of the SSL communication.

### 9.2.3 User Tools and EMail Gateway Plug-ins

As user tools are web-based, the connection between Web browsers and Commius systems will be encrypted and digitally signed (e.g., SSL with symmetric authentication and appliance of filters). The security of the email input depends on the protocol used by the user that sends the e-mail. The AA (Authentication and Authorization) in Commius can be related to SMTP-AUTH authentication if the user uses that protocol. We will perform the encryption of data for private data transfer (such as emails and user profiles) which implies the need of selecting a standard.

### 9.2.4 Security Connectors

Security Connectors are used via filters to ensure that Commius System Connectors will connect to existing/external systems in a secured way. As the external systems are fixed in Commius and cannot change dynamically, Commius must ensure the reliability before adding them to the repository and then the system could check their digital signatures or any other identity mechanism.

The definition of the connectors must include security parameters, such as the user that must be created inside each system, what kind of information will be extracted (confidential issues as Data Protection Law), encryption of the data, etc. Whenever possible, security will not be handled by the connector resident in each system. Instead, security should be handled by filters inside Commius system (as Authentication/Authorization filters). These filters will be configured in the application server including any needed parameter.

In the following there is a brief description of the type of connectors that will be present in the system, and how Security Connectors are envisioned:

- **Web Services**: Communication in web services is performed via XML over http/https, so it is viable to include filters that also provide support for security mechanisms such as WS-Trust or WS-Security
- **Database/file system access**: Because accessing directly to a database or a file system is an extremely exposed measure, it is highly recommended that these accesses are performed via web services instead. Anyway, should these accesses be mandatory provided by Commius, at least an encrypted connection (preferably SSL or TSL) should be enforced in the communication with these systems.
- **ESB (Enterprise Service Bus) and ERP**: Are likely to enforce its own security architecture. Whenever possible, the configuration with such systems should be configured to include encrypted connections, preferably SSL or TSL.
- **BPM Engines**: (workflows) usually rely their security issues on an external security provider, e.g., an active directory server or an LDAP server. Communication with
such servers should be configured to be performed by encrypted connections, preferably SSL or TSL.

- Payment mechanisms like Visa API enforce their own security mechanism. However, they are strong enough, thus Commius can trust them.

### 9.2.5 External security policies

In Commius system transmitting security policies is discouraged for the following reasons:

- All transmissions will be digitally signed. The source of each request will be uniquely identified. Usually a SME will grant access or not to its resources depending on the identity of the client, but not in its role, or, at least, not the role of an external organization.

- Transmitting roles might be subject to fraud. Because any people can install the Commius software, a hacker might install the software and grant himself a high level of permissions. Hence a PME cannot take any decision based on an external role because it is a problem of trust, not a problem of transmission security. Thus the only source of trust is based on authentication, and this is (supposedly) warranted by the digital signature.

- Transmitting security policies might force the system to evolve to a federated system to let the several Commius servers to access external authentication servers, and that would increase security complexity very much.

### 9.3 Security components

![Overview of security components](image_url)

**Figure 27: Overview of security components**
Figure 27 describes the component view of security layer. In this view, we distinguish between third party security components which are available for Commius and components to be developed in Commius. This view also includes various external systems such as databases, file systems, EPR, and Web services.

### 9.3.1 Third Party Security Components

Third party security components will form part of the Commius system but will not be developed by the Commius consortium. Instead, Commius will provide or ensure they are present and configure them. Though they will not be developed by the consortium, they are included in the diagram because many of the security measures will be performed through the parameterization of these modules, so we include it for clarifying that parameterization.

- **Authentication server (LDAP):** Each Commius system will need access to a local authentication server (for instance a LDAP server), which will be responsible for storing user attributes, like digital signatures, roles and other permissions. If there is not such an authentication server available in the PME, Commius should install an open source version and create at least a single user and associate him with the digital certificate of the enterprise/user. The availability of such a digital certificate is mandatory for COMMIUS. Such default user will be considered to have all the rights associated with the system.

- **Service Container:** The service container module, part of the Commius hosting environment, has the task to execute Commius modules which are exposed as services. *Filters* are installed as components of each application deployed on the web container, and many security parameters can be adjusted in the parameterization of the Service Container. Service containers can usually be parameterized by modifying some configuration files per application deployed. These files are accessed automatically by the *Parameterization Component* of the service container before launching any service. The Parameterization Component is a part of the Service Container. We include it in the diagram because many security parameters, like the use of SSL for transmission and the enforcement of double authentication can be (and usually are) configured this way. Any connection to the Commius system should be established over a SSL protocol, for instance with https (https is http over SSL) with a public key included. Besides, double authentication must be enforced because communication among several Commius applications might occur.

- **System encryption:** is part of the libraries provided by the Hosting Environment. It is exposed in the diagram to reflect that the encrypted persistence component will act as a wrapper for it.

- **DAO (Data Access Object):** Any implementation of the DAO design pattern which give access to any kind of datasources. It will be used to access data in external systems, like databases. The Hosting Environment provides this component, like data sources or queue systems, which allows to access external systems and lets them to be configured through configuration files. These components should be configured to ensure that encrypted transmission is enforced, normally by using SSL.
9.3.2 Commius-developed Security Components

9.3.2.1 Filters

Filters are the way in that Commius will perform such security operations that cannot be simply parameterized. The system will be open for further (and future) mechanisms by including a chain of filters that will be applied to access to the systems. Besides the SSL process of authentication, upon arrival (or exit) of a communication to/from the system, the chain of filters will be executed, altering the corresponding input or output, for example, by providing WS-SECURITY support. Including such chain of filters will allow further development of support for external systems. Among the filters included in the system, the final version of Commius will include a black list filter to prevent access to sites already classified as dangerous.

9.3.2.2 Encrypted persistence

All configuration files will be stored in an encrypted store and accessed only through the Commius configuration utilities, which will sign them with a Commius private certificate to avoid them to be hijacked by any attacker. Any other intermediate files generated by Commius must be stored encrypted, too, to prevent them from being altered by an attacker. Encrypted persistence is a wrapper common to all the Commius system to provide easy access to system encryption thus hiding other modules from the complexity of dealing with encryption.

9.3.2.3 User Management API:

With a similar philosophy to encrypted persistence, User Management API will be a wrapper common to all the Commius system that will provide access to user attributes, thus hiding other modules from the complexity of dealing with authentication systems and user tokens.

9.3.2.4 Security Classification

This component is responsible to give a security classification for a token from its type. It implements features discussed in Section 9.2.1.
9.4 Security/Privacy Control and Data Flows

Figure 28 describes a general sequence of filtered invocations. The user authenticates in the authentication server and makes a request to the system. The request performs operation and generates an output to another system. This output is filtered both in the original Commius system and in the second system, which processes it and sends a reply, which is filtered in both systems as well and finally notified to the user.

Figure 28: A general sequence of filtered invocations
Figure 29 describes the flow of a chain of filters. Whenever a message/request/response is delivered into or out of the system, a chain of filters is executed for that message. Each filter will be developed and configured by the Commius consortium.

![Flow of a chain of filters](image)

Figure 29: Flow of a chain of filters
Figure 30 describes a general sequence of access to the encrypted persistence component. To store a protected value, client classes will call the store method, which will be responsible for accessing the system encryption and store the value. To access a previously stored value, the get method is invoked, which is responsible for accessing the system encryption to get decrypt the value.

![Diagram](image-url)

**Figure 30**: A general sequence of access to the encrypted persistence component
10 External Systems

This section will discuss external systems to be supported and integrated with Commius infrastructure. External systems are those which are not part of the Commius but are utilized by the Commius system to provide necessary features for the user to conduct the business. External systems, therefore, comprise legacy systems but also other services. From the requirements, the following external systems need to be supported.

- SQL databases
- Spreadsheet Applications
- Web applications
- Fedit’s Document Repository
- Aitek’s proprietary electronic file system
- SingularLogic’s ERP,
- Softeco Sismat repository,
- Softeco ERP,
- Softeco proprietary job management system

The above-mentioned list of external systems is non-exhaustive. Therefore, Commius will be open to support other external systems by means of providing open interfaces and easy customization strategies.

An external system will be accessed from any components and modules within Commius through the System Connector which provides various plugins and mechanism to interface to external systems. Depending on particular external system, a different plugin can be implemented for accessing the external system or an existing plugin can be customized to work with the external system. Some external systems are described in detail in Table 1, Table 2, Table 3, and Table 4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Functionalities</th>
<th>Implementation and Deployment</th>
<th>Interfaces</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality assurance system</td>
<td>Provide features to store and retrieve documents with versioning and search functionalities</td>
<td>Running under Linux Red Hat, implemented in Oracle DB – Perl</td>
<td>Access through HTTP protocol, API with CGI GET/POST</td>
<td>HTTP BASIC authentication</td>
</tr>
</tbody>
</table>

Table 1: Overview of Aitek Document Repository

<table>
<thead>
<tr>
<th>Name</th>
<th>Functionalities</th>
<th>Implementation and Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONDA (Edisoft S.r.L)</td>
<td>Simple ERP quite common among SMEs in Italy.</td>
<td>Running in Windows (all platform supported), using COM/COM+ components</td>
</tr>
<tr>
<td></td>
<td>The main functionalities implemented are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Customer management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supplier management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accounting and Financial (purchase orders, invoice, bank reports..)</td>
<td></td>
</tr>
</tbody>
</table>

© Commius Consortium
Based on SQL database

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Direct access to Database tables import and export trough XML format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>User profiling and access with pass and login Privacy management according with Italian legislation</td>
</tr>
</tbody>
</table>

**Table 2: Overview of SOFTECO ERP system**

<table>
<thead>
<tr>
<th>Name</th>
<th>“Applicazione Commesse” Proprietary software developed by Softeco</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionalities</strong></td>
<td>Management and taking of company jobs: Quotations, Job open, budget, personnel costs monitoring, reporting, final report.</td>
</tr>
<tr>
<td><strong>Implementation and Deployment</strong></td>
<td>Running in Windows Based on SQL database</td>
</tr>
<tr>
<td><strong>Interfaces</strong></td>
<td>Direct access to Database tables</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>User profiling and access with pass and login</td>
</tr>
</tbody>
</table>

**Table 3: SOFTECO Company Job Management system**

<table>
<thead>
<tr>
<th>Name</th>
<th>Singular Enterprise (Sen)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionalities</strong></td>
<td>ERP with the following modules:</td>
</tr>
<tr>
<td></td>
<td>• Financial &amp; accounting</td>
</tr>
<tr>
<td></td>
<td>• Purchase</td>
</tr>
<tr>
<td></td>
<td>• Sales</td>
</tr>
<tr>
<td></td>
<td>• Production</td>
</tr>
<tr>
<td></td>
<td>• Warehouse/ distribution/ logistics</td>
</tr>
<tr>
<td></td>
<td>• Reporting</td>
</tr>
<tr>
<td><strong>Implementation and Deployment</strong></td>
<td>Developed in Delphi Running in Windows Based on Oracle database</td>
</tr>
<tr>
<td><strong>Interfaces</strong></td>
<td>APIs / web services (limited to specific functionalities)import and export trough XML format</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>User profiling and access with login &amp; password Access rights based on user’s profile</td>
</tr>
</tbody>
</table>

**Table 4: SingularLogic ERP**

The above-mentioned external systems are just exemplary. The concept of external systems in Commius potentially allows us to integrate any types of external systems which follow the request-response model. One potential is to use System Connectors to integrate Web service-based systems for enterprise interoperability and collaboration. This would further enhance the integration of email-based business solutions with Web service-based ones.
11 Customizations and Deployments

In this section, we discuss different customization and deployment possibilities, including individual Commius, network of collaborative Commius-enabled SMEs, Commius service provider, and client-side extension.

11.1 Individual Commius

An SME can utilize Commius as an individual utility, only as a single utility for the SME without a connection to other Commius-enabled SMEs. In this case, the deployment of Commius software architecture will include all core parts of the Commius (the Email Gateway Plugin, Modules, System/Semantic/Process Interoperability Components, Privacy and Security Components, and Data Management Components) and will be configured to work with external systems decided by particular deployment.

The Commius core system can be deployed and managed by a specific SME for its business or by a service provider for leasing. The deployment of Commius will follow typical procedure in which many features can be configured and updated during the operation of the Commius. In particular, business modules will be easily added and the whole Commius system can be configured based on a Web portal.

11.2 Network of Cooperative SMEs

The Commius also supports a network of cooperative SMEs. This can be achieved by configuring individual Commius systems to a trusted network of cooperative SMEs. By trusted network, we mean that SMEs have some kinds of business agreements in their cooperation. Based on such business agreements, technical agreements and configurations can be established to produce a trusted network, for example, using common PKI infrastructure or sharing user access control. SMEs in the trusted network might or might not be Commius-enabled. Depending on the trusted network, cooperative SMEs may share common document and business process templates which might be stored at individual SMEs or at a shared repository.

In the following, we analyze the structure of the Commius cooperative network based on centralized and peer-to-peer models. The main purposes of the network of cooperative Commius systems are (i) to enable companies to exchange documents more easily by ensuring that document they receive in e-mail are automatically mapped to a set of document types defined internally by the company and (ii) to enable companies in the same network to cooperate in answering client requests (e.g., the partner search in a network of Commius-based SMEs described in Section 7.3).

Figure 31 describes a high-level view of a network of cooperative Commius-based SMEs. Commius installations can be hosted in the company infrastructure or be provided by a service provider, as discussed in the previous section. All companies in the same network have some agreements which enable them to cooperate by using Commius. Based on such agreements, security and privacy policies are established to ensure that a company can send some requests to another company. The trusted Commius network means that such policies
are established between companies in the same network of cooperative Commius-based SMEs. With respect to trusted network, we distinguish two types of Commius networks configuration:

- highly trusted Commius network, e.g., between departments of the same companies
- not highly trusted Commius network

For highly trusted Commius network, data might be synchronized and centralized model can be applied. For example, when Commius network is established between different departments of the same company, like the case of FeDIT, customers data can be synchronized and shared to improve the quality of Commius suggestions. On the other hand, when the trust in the network is low, P2P cooperative activities can be used. For example, a company A will handle requests from company B in a similar manner like a request from its normal customer but with some privilege services. This will be decided, for example, based on rules.

![Figure 31: A network of cooperative Commius-based SMEs](image)

### 11.2.1 Centralized Shared Resource Systems

When agreed, companies in the same network can establish a shared resource system which can be used to host shared documents. Such a shared resource system can be considered as an external system to a Commius. Therefore, the shared resource system can be accessed by any Commius in the network via the System Connector. The shared resource system can be hosted by an SME in the network or by a service provider and will operate under agreed security and privacy policies.

In this model, the shared resource system will support the traditional and well established approach to the exchange documents/information by mandating a set of standard documents which should be used when communicating between the companies. The basic elements of this approach are:

- A centralised set of standard document types.
- At each company, localised mappings between their existing document types and the
standardised set.

Figure 32 describes a simple scenario in which different companies can request and upload shared documents using the shared resource system. A requester can send an email to the Commius A to ask for a document. The Commius A considers the shared resource system as an external system, thus invoking the System Connector to obtain the document which is returned to the requester. In another side, the SME B can upload a document to share it with other companies in the same network. Again a user with in SME B will upload the document to the Commius B which knows that the document should be stored into the shared resource system.

Of course, a company can also completely replace its internal document types with those mandated in the standard. Once a set of standard documents has been agreed on, set up and is used by everyone within a grouping then the above approach is ideal.

11.2.2 Peer-to-Peer Shared Documents

The peer-to-peer shared documents model is trying to provide something approaching the centralized model for individual SME's. In such a scenario we cannot assume that everyone communicating with the company will follow specific formats or indeed that they use Commius. This motivated our proposed approach.

The basic idea behind this approach is to focus on the information within the documents rather than their actual syntax. To do this it features:

- A set of standardised tokens, each representing an atomic item of information which can be found within a document
- A set of rules for pulling these tokens out of documents
• A set of local document types defined in terms of the information tokens

The basic approach is then to take an incoming document, identify what information tokens it contains and hence identify the closest matching local document type(s) and insert the information into these. In the peer-to-peer document sharing, each Commius deployment includes an initial set of information tokens and rules for identifying these. Each company then uses these tokens to define its local document types. When a company wants to retrieve a shared document from the network, the Commius of the company sends a request via email. Upon receiving the email, the Commius of a company can use rules and tokens to find relevant documents and propose the result which can be accepted by a user and sent to the sender.

If a company later finds out that it needs to add new information tokens or extraction rules or document types then it simply does so and uses the updated set of tokens/extraction rules in its local operations.

The information contained in the new tokens/extraction rule remains at a local level while it could be of interest to other companies of a Commius network. This set up allows every individual company to change the set of token and documents types that they use. Instead, it may prove preferable to coordinate the set of tokens/document types used within a network of cooperative companies. In this case the centralized model can be used to share their document types/information tokens. In such a scenario the idea isn't to share every new document type or information token discovered. Instead individual companies propagate those additions they think that their partners will find most useful.

Within this network of cooperative SMEs, an information token is an element of UNCefact core components standard library. Thus, for example, it might be an address. Commius will be provided with a basic set of these together with rules for identifying them.

11.2.3 Peer-to-Peer Model for Cooperative Activities

The peer-to-peer model for cooperative activities supports companies in the same network to assist each other in answering requests from the user. When a user of a company sends a request, the company Commius system might not be able to solve the request. In this case, the company can use the cooperative network to find another company which can solve the request. One example of such a scenario is the partner search in a network of Commius described in Section 7.3.
Figure 33: A simplified example of a cooperative activity in answering a request

The peer-to-peer model for cooperative activities works on the assumption that each Commius deployment includes rules for identifying when a request should be forwarded to another company. Such rules are updated over the time based on business exchange within the network. Rules can be used to (semi-)automatically forward emails or to create a new request based on an existing one. A company receiving a request from another company in the network might treat the request in a different way with a request from its own user.

Figure 33 describes the simplified example of the above-mentioned scenario based on the partner search example given in Section 7.3.

11.3 Commius Service Providers

Based on facilities provide by Commius, some service providers might become “Commius service provider” to support SMEs doing business with Commius-enabled solutions. The Commius architecture is designed in a generic way so that one can utilize Commius to support different Commius service providers models. A Commius service provider could provide
• Security for a trusted network of cooperative SMEs.
• External services for managing and sharing templates
• Commius-enabled systems under SaaS
• Etc.

A concrete type of Commius service provider requires a different type of configuration. For example, a provider may utilize the centralized model mentioned in Section 11.2.1 to setup a common, centralized shared repository for document templates, or the P2P model mentioned in Section 11.2.2 to provide a P2P-based Commius for a set of companies.

11.4 Client-side Extension

In the current version, the deployment of Commius software architecture is envisaged as server application handling email communication within an enterprise, but Commius solution can be easily deployed on client-side machine to process client’s personal emails and thus ensure security of valuable information. We discuss this issue here and technical detail solutions for client-side extension will be studied in WP4.

<table>
<thead>
<tr>
<th>Server-side application</th>
<th>Client-side application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server application must be trusted by hosting client.</td>
<td>User has full control over his personal and email data.</td>
</tr>
<tr>
<td>Users can easily share knowledge using collaborative modules.</td>
<td>User has only restricted knowledge sharing functionality.</td>
</tr>
<tr>
<td>Suitable for enterprise email account.</td>
<td>Suitable trustful email communication, personal use or testing.</td>
</tr>
<tr>
<td>Requires web server.</td>
<td>Web server is optional.</td>
</tr>
</tbody>
</table>

Table 4: Differences in Commius server-side and client-side deployment.

![Figure 34: Server-side and client-side models](image-url)
Client-side applications can communicate with the Commius system the same way as server-side with only one difference that the web browser can open user interface from file and do not require any web server installed. Opened web page communicates with locally running Commius modules using Google Web Toolkit (GWT) implementation of RPC. The communication is sketched in Figure 28 between “Data Services”, “GUI Processor” and “Web Browser”, where “GUI Processor” represents generated HTML page with initialization JavaScript code and “Data Services” represents data feeds handled by GWT JavaScript code running in “Web Browser”. Note that due to security issues, some modules may have restricted functionality in the client-side application deployment. The use of scripting technologies like Javascript and AJAX will be carefully investigated due to security concerns.

23 Here are some references about why scripting technologies can lead to security holes:
http://groups.google.com/group/Google-Web-Toolkit/web/security-for-gwt-applications
12 Implementation Roadmap

In this section, we will describe our implementation roadmap. We first analyze existing technologies. Based on that, we propose the roadmap for the implementation.

12.1 Analysis of existing technologies for Networked Enterprises

12.1.1 Enterprise Service Bus (ESB)

An ESB is a software infrastructure that provides solutions for integrating heterogeneous software systems\(^{24}\). Typically most ESBs support the fundamental message transformation, routing and mediator but a few of ESB products also support workflow engines and SCA containers\(^{25}\). Most ESB systems use XML as the main communication protocol and support Web services. The main advantage of ESB systems is that they support the integration of enterprise applications via through messaging systems. Thus, different applications can be integrated through loosely coupling models and common data exchange representations.

In our view ESB would be suitable for the integration of legacy systems which are diverse. In this sense, a Commius Message Connector can be interfaced to an ESB system which in turns is responsible for the integration of various other systems. This would enhance the extensibility of Commius can support (since new services/applications can be integrated via ESB) as well as simplify the security issue when integrating Commius with other systems.

12.1.2 Web Services

The primary idea behind Web service technology is to enable a system to be composed into a set of independent components which execute in a distributed manner but are tied together into a coherent system by passing appropriate messages between them. Web Services technologies simplify the integration of different services provided by different organizations by introducing well-defined interfaces and protocols.

Since the Commius architecture is designed to contain a set of independent components tied together by a messaging system Web services form a potential paradigm for usage within the system. However there is a very important factor which must be considered here:

- The Commius system is designed to be a single piece of software which is downloaded and then run on a local machine.
- Since the primary motivation behind web services was to support distributed execution this suggests that systems designed to provide modular architectures on single computers will be more appropriate within this context.

Web services technologies can be used to expose some Commius modules, thus the modules can be accessed from remote site.

---


12.1.3 Service Composition Architecture (SCA)

SCA\textsuperscript{26} (Service Component Architecture) provides a programming model for building applications based on a SOA (Service Oriented Architecture). The idea of SCA is to have a series of business logic functions which, assembled together, provide a solution to a particular business need. SCA provides a model for the creation of service components as well as their integration and composition. SCA aims to encompass the widest range of possible technologies, both for implementation of the components and their interaction methods. Therefore, it supports the composition of services built based on different technologies such as Web services, OSGi, Messaging systems and Remote Procedure Call (RPC). One open source of SCA frameworks is Apache Tuscany\textsuperscript{27}. In SCA, a service implementation offers a concrete business function and can be implemented by using different technologies such as Web services, OSGi, and BPEL. A service can be built based on other services, service references and properties through the assembly process.

With regard to Commius, SCA may be the choice in composing and integrating the higher-level modules of the architecture and external systems integration. SCA is also suitable for developing Commius modules which compose Commius components.

12.1.4 OSGi

The OSGi (Open Services Gateway Initiative)\textsuperscript{28} technology supports the development of dynamic module systems based on Java. To this end, the OSGi framework, the core of OSGi technology, provides a standardized environment addressing issues related to execution environment, modules, life cycle management, and service registry of software components. On top of the OSGi framework, the OSGi Alliance has specified many services. In the same OSGi platform, clients can find services via a registry. Furthermore, OSGi components can directly react on the appearance and disappearance of other OSGi components. OSGi services can be exposed as Web services as well. There are these abstract types of services such as Framework Services, System Services, and Protocol Services. Various implementations of OSGi are available such as Knopflerfish\textsuperscript{29}, Equinox\textsuperscript{30} and Apache Felix\textsuperscript{31}.

OSGi model seems very suitable for the development of Commius architecture. In particular, OSGi execution environment is based on Java and very lightweight, making it suitable for the development of the core of Commius system of which components are hosted in a centralized environment. The concept of OSGi bundles match to Commius components and modules. OSGi supports registry mechanism which can be used to develop the modules discovery in Commius.

12.2 Commius hosting environment and programming models

Because Commius components and modules need to be lightweight and plugin-able and because we also need to support Web interaction, we select OSGi as the main platform for

\textsuperscript{26} http://oasis-openocs.org/
\textsuperscript{27} http://tuscany.apache.org/home.html
\textsuperscript{28} http://www.osgi.org/Main/HomePage
\textsuperscript{29} http://www.knopflerfish.org/
\textsuperscript{30} http://www.eclipse.org/equinox
\textsuperscript{31} http://felix.apache.org/site/index.html
Commius software development. OSGi will not only allow lightweight components and modules to be developed but also support easy installation and management. Furthermore, we can also expose OSGi-based modules to Web services.

12.3 The Roadmap for the Implementation of Commius Architecture

In order to start the prototype of the Commius architecture, we propose the following roadmap

- Selection of technologies: considering the above-mentioned technologies and the requirements of Commius, we are currently utilizing Apache Felix\[^{32}\] to implement the core of Commius.
- Selection of programming languages and platforms: to support the easy installation and platform-independent Commius, the core Commius will be developed atop Java-based hosting environment so that Commius can be easily deployed in various platforms. We use Linux based system for mail servers
- The Email Gateway plugin, module management and system-oriented modules, system interoperability layer will be developed in WP4.
- The semantic interoperability components will be developed in WP5
- The process interoperability components will be developed in WP6
- The business modules, including those for the support of network of cooperative SMEs, and tools for end-users will be developed in WP7
- Tools and techniques for customizing Commius will be addressed in appropriate WPs, such as WP4, WP5, WP6, and WP7.
- Demonstrations and integration will be carried out in WP8 and WP9.

[^{32}]: http://felix.apache.org/
13 Conclusions

In this deliverable, we have described the final version of the Commius architecture. We defined common terminologies and concepts for the Commius architecture and designed major components and modules for realizing the Commius Interoperability Service Utility. External systems to be supported have been identified and possible customization and deployments of Commius are presented. We have also analyzed existing technologies which could be used for the implementation of the Commius prototype and provide guidelines for the implementation. The Commius architecture is designed as an open, secured and customizable system, supporting networks of cooperative SMEs to perform their daily business based on email.

The following items have to be addressed in corresponding WPs.

- Specification and implementation of modules: including interfaces, semantics of input and output
- Specification and implementation of components
- Assurance of security and privacy issues.
- Dynamic configuration, detailed customization and deployment methods for network of cooperative SMEs.