	<p><b>IST-2001-34340</b></p> <p><b>EUPKI- WP6</b></p> <p><b>D6.1 Integration Progress Report One</b></p>
<p>Distribution List :</p>	<p>Project Partners</p>
<p>Author :</p>	<p>Lionel Prades, Maxime de Jabrun, Peter Sylvester <b>Groupe ON-X</b></p>
<p></p>	<p>Eric Choffat <b>EADS</b></p>
<p>Distribution List :</p>	<p>Project Partners</p>
<p>Authorised by :</p>	<p>Paul-André PAYS</p>
<p>Date of Issue :</p>	<p>November 28<sup>th</sup> , 2002</p>
<p>Issue :</p>	<p>0.5</p>
<p>File Name :</p>	<p>EUPKI-WP6-D6.1-V1.0.doc</p>
<p>Work Package :</p>	<p>WP6 Integration</p>
<p>Deliverable Number :</p>	<p>D 6.1</p>
<p>Deliverable Type :</p>	<p>Preliminary Draft</p>
<p>Deliverable Nature :</p>	<p>Integration progress report one</p>
<p>Total Number of Pages :</p>	<p>18</p>
<p>Contact Details for EUPKI :</p>	<p>Project Coordinator Yann FRAVAL GIP-MDS</p>
<p></p>	<p>mail : <a href="mailto:yann.fraval@gip-mds.fr">yann.fraval@gip-mds.fr</a></p>
<p></p>	<p>web site : <a href="http://www.eupki.org">www.eupki.org</a></p>

## 0 Table Of Contents

<b>0</b>	<b>TABLE OF CONTENTS .....</b>	<b>2</b>
<b>1</b>	<b>DOCUMENT CONTROL .....</b>	<b>3</b>
1.1	ABSTRACT .....	3
1.2	KEYWORDS.....	3
<b>2</b>	<b>MANAGEMENT OVERVIEW .....</b>	<b>4</b>
2.1	EXECUTIVE SUMMARY .....	4
2.2	SCOPE STATEMENT .....	4
<b>3</b>	<b>PREPARATORY ACTIVITIES OVERVIEW.....</b>	<b>5</b>
3.1	INTEGRATION CONCERNS AND COLLABORATION ENVIRONMENT .....	5
3.2	ACTION PLAN OVERVIEW.....	5
<b>4</b>	<b>INTRODUCTION AND GLOSSARY.....</b>	<b>7</b>
4.1	GLOSSARY .....	7
<b>5</b>	<b>CONTEXT DESCRIPTION.....</b>	<b>8</b>
5.1	PROJECT ORGANISATION .....	8
5.2	EVALUATION OBJECTIVE.....	8
5.3	DEVELOPMENT METHODOLOGY AND COLLABORATION ORGANISATION.....	8
5.4	STATE OF THE ART .....	9
5.5	COLLABORATION ENVIRONMENT.....	9
<b>0</b>	<b>ANNEX A : COLLABORATION PROCESS.....</b>	<b>10</b>
0.1	OVERVIEW .....	10
0.2	ACTORS .....	11
<b>0</b>	<b>ANNEX B: INTEGRATION GLOBAL PROCESS PRINCIPLES.....</b>	<b>12</b>
<b>0</b>	<b>ANNEX C: STATE OF THE ART .....</b>	<b>13</b>
0.1	PROJECTS.....	13
0.1.1	<i>Apache</i> .....	13
0.1.2	<i>OpenSSL</i> .....	13
0.1.3	<i>NewPki</i> .....	14
0.2	TOOLS.....	14
0.2.1	<i>Record keeping</i> .....	14
0.2.2	<i>CVS</i> .....	14
0.2.3	<i>RT</i> .....	15
0.2.4	<i>Doxygen</i> .....	15
0.2.5	<i>Mailing lists</i> .....	15
0.2.6	<i>HTTP/FTP distribution</i> .....	15
0.2.7	<i>Gnu autoconf</i> .....	15
0.2.8	<i>Coding conventions</i> .....	15
<b>0</b>	<b>ANNEX D: COLLABORATION ENVIRONMENT OVERVIEW.....</b>	<b>16</b>
0.1	PUBLIC WEB SITE .....	16
0.2	THE PUBLIC FTP SERVER .....	16
0.3	THE PUBLIC MAILING LIST SERVER.....	16
0.4	PRIVATE WEB SITE .....	17
0.5	PRIVATE SOURCE REPOSITORY .....	17
0.6	PUBLIC SOURCE REPOSITORY .....	17

## 1 Document Control

<i>Issue</i>	<i>Date of Issue</i>	<i>Comments</i>
0.1	10 July 2002	Creation, internal working version ON-X
0.2	18 September 2002	Initial version containing global canvass + specific sections for the following items: common criteria analysis, state of the art of other projects, proposals for development methodology, organization, and collaboration environments, reviewed by ON-X and EADS.
0.3	9 October 2002	Document cleaning: <ul style="list-style-type: none"> <li>- Removal of Annex C, Common Criteria (cf. WP8 for detail)</li> <li>- Extraction of Annex A and G and integration in documents R6.1 and R6.2</li> </ul>
0.4	25 October 2002	EADS <-> ON-X meeting comments integration
0.5	5 November 2002	Submitted to the partners for comments.
1.0	2002-11-28	Approved

### 1.1 Abstract

The purpose of the deliverable D6.1 "Integration Progress Report One" is to make available a synthesis of the work done for the guidelines and rules, for setting up the collaboration server and tools.

This document will progress and be updated.

### 1.2 Keywords

EUPKI                      EUPKI, the libre software Public Key Infrastructure (project name)  
 WP6                      Work Package 6  
 D6.1                      Reference to the deliverable D6.1 "Integration Progress Report One"

## 2 Management Overview

### 2.1 Executive Summary

This document contains the synthesis of the preparation work to integration work package.

It points towards the formal description of the integration processes and the basis that underlay it.

It addresses flows and tools used for the whole integration of EUPKI.

### 2.2 Scope Statement

The scope of this document is to resume the different works realized during preparation of integration phase including different documents concerning it as annexes (or referenced in annexes).

Reference	Document
R6.1	Integration Process Overview
R6.2	Integration guidelines and acceptance procedures
R6.3	Collaboration Environment Overview

### 3 Preparatory activities overview

#### 3.1 Integration concerns and Collaboration environment

WP6 is has raised the following questions:

- Is the development model using early integration technically possible?
- How will an alpha version look like?
- When and how do we set up the integration environment?
- Details of CVS definitions and cooperation

They are addressed in three new reports: R6.1, R6.2 and R6.3. These documents are submitted for comments and approval to the technical board.

#### 3.2 Action Plan overview

The present document (D6.1) is the first of a three “Integration report” series. The second report (D6.2) is scheduled for the preliminary version (PV) along with the delivery to the partners of the version itself. The third report (D6.3) is due for the final version along with the delivery of the final version of the software and related documentation.

In order to enhance short-term efficiency and to assure best results in production and quality, WP6 is engaged in producing additional documents.

R6.1 “Integration Process Overview” describes WP6’s proposal for Integration Process. Next document, details this process.

The following tasks will be achieved in R6.2 “Integration guidelines and acceptance procedures”.

- Integration Procedure:
  - Validation of existing building blocks and tools.
  - Upgrade management for existing building blocks and tools.
  - Acceptance procedures (alpha / beta / release transitions).
- Development guidelines
  - Specification of documentation format and selection of associated tools (for instance, API documentation with Javadoc and Doxygen).
  - Coding language requirements (API coded in C, limit number of languages: scripts coded in Perl or PHP, interfaces in C++, etc.).
  - Development and integration tools requirements: Doxygen, autoconf, unitary testing: CUnit, CPPUnit, ...
  - Repository build up process
- Definition of Reports and Deliveries:
  - Development deliveries (i.e. input deliveries), which correspond to deliveries of source code and documentation made from development to integration (typically, from WP5 to WP6)
  - Distribution deliveries (i.e. output deliveries), which correspond to deliveries made from integration to testers and users (typically, from WP6 to WP7).

- Integration and Upgrade request report for existing building blocks and tools
- Acceptance report Template for development package refusal from integration.
- Integration Request report
- Release/Patch report Template (reference, new features, known bugs, fixed bug, CVS Tag), pre filled by Development, completed by Integration.
- Validation, Bug, Modification and Enhancement reports templates (testers/users).

Simultaneously, R6.3 “Collaboration Environment Overview will specify requirements (almost done) and constraints on collaborative servers.

The middle term following action, “Identification of tester and definition of its role”, will be achieved by first production from WP5.

WP5 is proposing to the Technical Board coding language rules by the end of November 2002. These rules must achieve the following requirements:

- Avoiding naming conflicts between modules and also for external contributions;
- Allowing object wrapable coding.

## 4 Introduction and Glossary

This document is for now divided into two parts.

The first one is the structure proposal for this deliverable as defined in the DoW, i.e. a synthesis of several aspects concerning integration.

The second one is a collection of appendices with detailed analysis or proposals to collect different points which request for comments and advices.

### 4.1 Glossary

<b>WP</b>	Work Package
<b>PK</b>	Software Package
<b>CC</b>	Common Criteria
<b>EAL</b>	Evaluation Assurance Level
<b>PP</b>	Protection Profile
<b>ST</b>	Security Target
<b>TOE</b>	Target of Evaluation
<b>TSF</b>	TOE Security Functions
<b>CM</b>	Configuration Management
<b>DoW</b>	Description of Work

## 5 Context description

The EU-PKI solution may integrate contributions from different sources. The main purpose of work package 6 is to ensure the availability of consolidated versions –with that appropriate level of quality that would enable to project to «label» them as official project approved released versions.

Several questions need to be addressed:

- What is the EUPKI organisation and how actors of the project collaborate?
- What can be done to the common criteria evaluation objective?
- What are the overall integration methodology and the procedures to be used among integrators, contributors, and users?
- What can we learn from other projects?
- What is the technical environment?

### 5.1 Project Organisation

In order to be able to define the global integration process we must clearly assign responsibilities of the project actors and how they collaborate with each other.

Annex 1 of the contract: “Description of Work” gives the general project organization and responsibilities for each work package, but no other document aims at describing how these actors will interact with each other in an iterative development process and how they will work with external actors.

*A proposal of the global development procedure, its actors, responsibilities and interactions is given in the documents R6.1 “Integration Process Overview”.and R6.2 “Integration guidelines and acceptance procedures”.Annex A resumes this approach..*

### 5.2 Evaluation objective

In order to comfort the future acceptance of EU-PKI, care should be given to the preparation and facilitation of any eventual CC evaluation.

This evaluation driver leads us to choose an Evaluation Assurance Level for the product (and therefore for integrated modules), in accordance with the future context of diffusion and usage; it let us propose to work keeping in mind the rules of EAL 2.

The main purpose of the analysis is to provide a software version that contains the necessary components and structure as required for EAL2.

### 5.3 Development methodology and collaboration organisation

EAL provide us with criteria and procedures for external releasing, not for our internal exchanges and procedures we will need to specify and to manage anyway.

There should not be any difference in modules handling, be Internal modules or external contributions, so goals of this document are to propose global guidelines which underlay the whole process of integration.

It is therefore necessary to describe a life cycle process of the software development process. Following many good experiences we propose a model and a process that can be characterized as 'integration before module implementation' containing the characteristics of quick prototyping.

The proposal is to make available as soon as possible an integrated version of empty or almost empty modules, in order to verify the overall high-level to low-level decomposition and validate user interfaces. This allows involving integrators and users as soon as possible in the development process and easily determining eventual missing functionalities of modules.

*Presentation of a proposal for the process, its interfaces, flows and the conditions of work are available as proposal in the document "Integration guidelines and acceptance procedures".*

***Annex B resumes this proposal.***

#### **5.4 State of the art**

Other instances of "libre" software projects can be observed for management modes and tools.

The most important observation is that other projects are beyond the state of initial development, i.e., they are essentially in a state of maintenance phase. Thus, there are no directly available procedures for integration.

Nevertheless, historical experience concerning new features shows that the initial development phases are sometimes relatively fast, and the project passes pretty fast to a semi-maintenance procedure.

***Cf Annex C for a more detailed analysis.***

#### **5.5 Collaboration environment**

First objective of the project is to release a PKI product.

Care must be taken not to invest too much budget in development of a "project management software", it is not the objective.

Then, it seems desirable not to use different technical means during integration and after release maintenance. Following the proposal for the integration procedures, the alpha version of the product will be integrated very early, and the beta versions can be considered as maintenance releases.

The tool that is used today in most similar projects is based on CVS.

*Basic requirements and constraints for the collaborative server have to be defined.*

***For proposal, cf Annex D and document "Collaboration Environment Overview".***

## 0 Annex A : Collaboration process

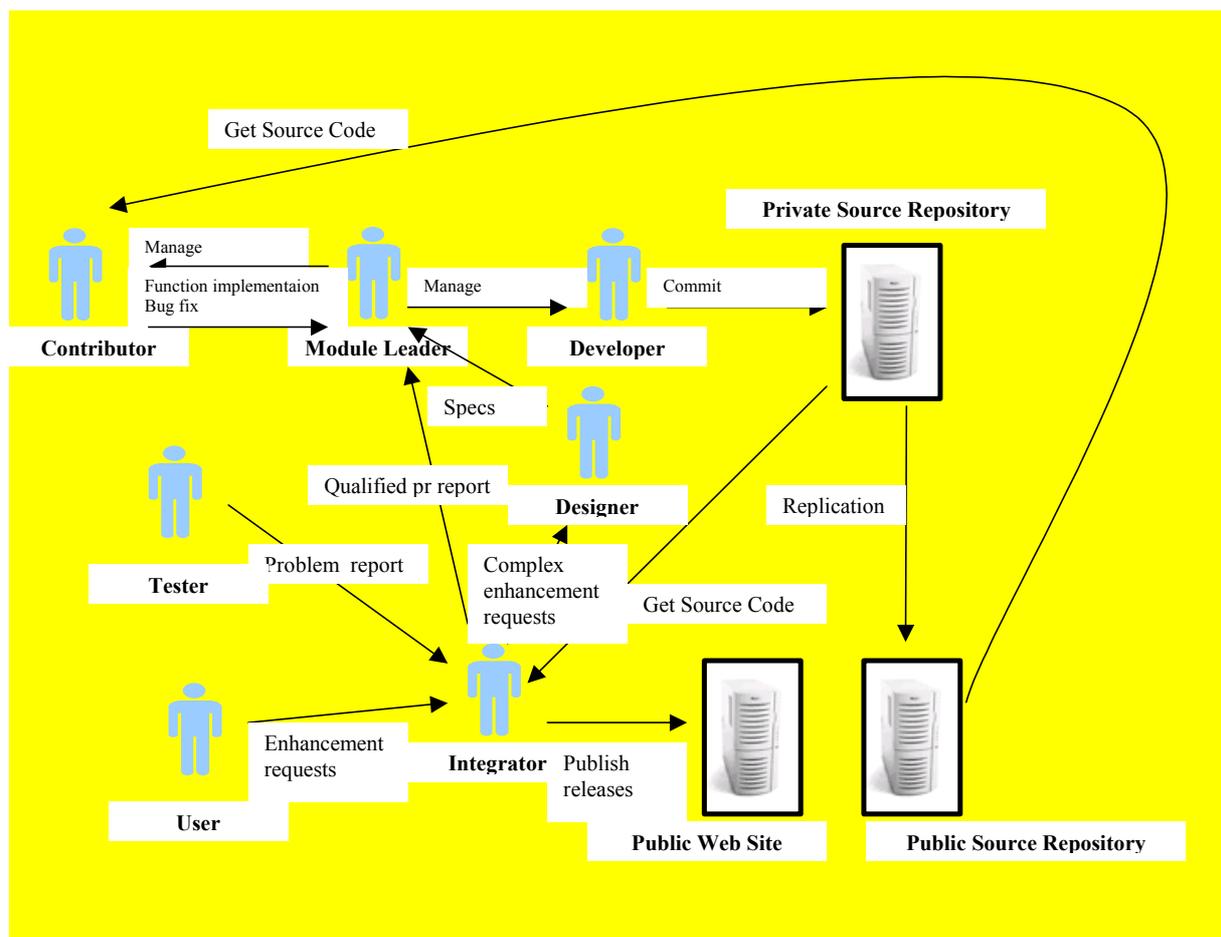
This annex summarizes the global “Integration Process Overview”, document to be produced.

A more specific “Integration guidelines and acceptance procedures”. See R6.2 1) presents the “integration roles and duties” and 2) specifies the external interfaces (the set of applicable rules and procedures) with mostly developers (strict rules are to be established and controlled) and users or testers.

In this chapter we make a proposal on how internal and external actors collaborate in the development process. Please refer to R6.1 and R6.2 for more details.

### 0.1 Overview

The following diagram resumes the essential interactions between project actors in an iterative development process.



Integrator receives contributions from development or exterior. It checks their quality (acceptance procedure and validation of external contribution). Then WP6 integrates

them and performs global testing of the integrated product. When users requirements are estimated completed, WP6 publish a “stable” version of the product. WP6 is also responsible for the support for external users and external testers of the product. Each step is ruled through an interface defining requirements for related actors. These interfaces are referred to in annex G.

Private source repository, Public source repository and Public Web Site are described in Annex S and will be detailed in R6.3 “Collaboration Environment Overview.

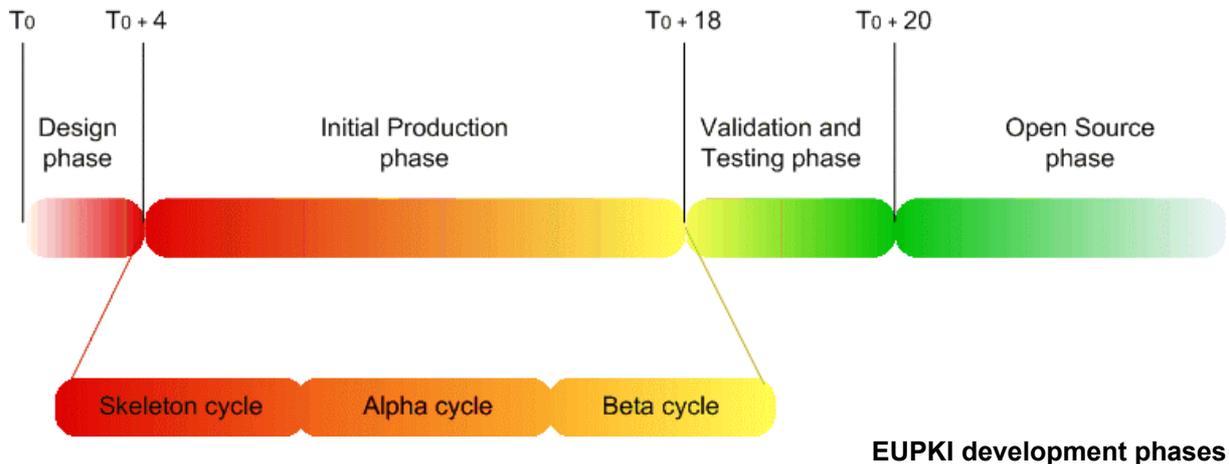
## 0.2 Actors

We define the following actors:

- **Designer (WP4)**
  - Designers specify at the overall project level the complex workflows and functions.
  - They have to translate the functions coming from the requirements (a functional specification) into lower level specifications.
- **Integrator (WP6)**
  - Integrators consolidate developers and external contributions so as to produce a functional product for publication and testing.
- **Module Leader (WP5)**
  - The module leader is a developer who is in charge of development activities around one particular module. Module leader is a member of WP5 and is nominated by developers.
- **Developer (WP5)**
  - The developer is in charge of implementing what have been specified. He is a member of the consortium and has write access to the source code repository. In the future, valuable contributors may become developers upon Steering Committee agreement.
- **Contributor**
  - Contributors are external persons who may furnish bug fixes (patches) or a complete function implementation; they may also participate to integration work or document redaction. They are trained to EUPKI development process by the Module Leader (development contributor) or WP6 leader (integration contributors). Contributors must follow EUPKI procedures and constraints.
- **Tester**
  - Testers are the persons who test alpha and beta versions.
- **User**
  - Users are the persons who use the products of the Project. They are typically system integrators or PKI Service Providers. WP3 members fall into this category but we hope EUPKI will have of a lot of external users.

## 0 Annex B: Integration global process principles

As for the Initial Production phase of EUPKI, a three cycles integration process is proposed for each block of EUPKI (CA, RA, GUI ...). This approach is thought to allow the integration phase (WP6) to start as early as possible with the core activity of integration tasks.



**Skeleton** cycle integration uses the simplest implementation of internal APIs to validate that packages can be integrated and to legitimate high and low level design first instances and so the chosen structures.

**Alpha** cycle attends to deliver all functions required to run a particular end-user scenario. Other side functions may be simulated.

**Beta** cycle aims at providing a **fully tested implementation** of functionalities and security functions, and the whole associated documentation for public diffusion.

Finally, when beta cycle is completed, product is released as Open Source.

Each cycle is iterative. When cycle requirements are achieved, WP6 proposes to the Technical Board to validate the beginning of next cycle.

For a deeper description and analysis of this process proposal, please refer to

- R6.1 "Integration Process Overview"
- R6.2 "Integration guidelines and acceptance procedures"

## 0 Annex C: State of the art

Other instances of “libre” software projects can be observed for management modes and tools. We look at some actual projects and some tools in use. Although they give some basic rules to establish the development environment, they only give limited ideas on how to organize an initial development and integration.

### 0.1 Projects

Making a detailed analysis of other open source projects is not an objective of the project. We just list a few important projects whose results are well known and characterize how they are organized.

As we will see, the organizational part highly depends on the good will and the competence of the core developers and integrators. Furthermore most projects are already in a state that is beyond the initial integration, the actual organization of the project actually corresponds to the need for continuous maintenance.

Furthermore, the processes are still evolving in time as a function of the project growth.

#### 0.1.1 Apache

This project is actually a collection of several sub-projects. There are more than 50 developers who also act as integrators. The development is organized inside the Apache Software Foundation, only members of the foundation act as integrators. Becoming a member is by invitation and voting. The members also act as developers but contributions from the outside are possible and welcome, in particular to address portability issues.

The developers use a CVS environment to organize the work on different pieces.

Details of the procedures are documented.

Code development in C must follow some rules that are also documented.

The actual development environment is available to everybody through CVS. Intermediate releases are generated and distributed through HTTP and FTP in irregular intervals, motivated by important development cycles or security fixes.

#### 0.1.2 OpenSSL

The structure of development environment of openssl is characterized by the history of the project. It is not formally defined according to some criteria but a result of the needs of the developers and the users.

The development environment consists of a CVS service controlled in an ad-hoc way by the small numbers of principle developers who, in our terminology, also act as integrators.

The CVS environment is available to everybody. In order to simplify the usage and testing in other environments, daily snapshots are available through FTP/HTTP.

In irregular intervals, minor and major releases are provided using several beta type fixing in order to allow portability testing by other people since a large number of platforms is not available to the core developers.

The daily snapshots also provide a simple method to recover from errors committed by the core developers, as the correction cycle is about one day.

About a year ago, the number of external contributions including error reports has reached a state where the ad-hoc treatment by the core developers created too many errors, loss of contributions, etc. This had lead to the usage of rt as an incident tracking tool. All external contributions and error notifications are handled now through rt.

All important components of openssl have there module level test routines; for each cryptographic algorithm there is a set of test vectors, and the actual implementation is tested against them (using make tests);

The usage of CVS was introduced when the original Ssleay code taken over by OpenSSL project. The code has a few problems, in particular since the include files and the test directory is generated either as symbolic links or as copies.

The portability configuration does not use the GMU autoconf technique, the actual portability testing is a historical result.

### 0.1.3 *NewPki*

This is a recent development that addresses a similar objective as EU-PKI. The software is actually developed and managed by one person. The interesting point is the way how tools are used.

All software development is organized through CVS. The C and C++ library are only available through CVS.

There is a generated HTTP tree representing the java structures.

Furthermore all documentation is generated using **doxygen**.

## 0.2 **Tools**

We give a list of tools that are used in other open source projects as management and development tools.

We note that most of the tools are mainly useful either as general quality mechanism or to simply maintenance issues.

### 0.2.1 *Record keeping*

There is a simple but efficient means to document the progress of the software and maintenance activities. It consists of simply summarizing all activities in a single CHANGES file.

### 0.2.2 *CVS*

In most projects the actual distributed development is organized by CVS. In a distributed environment the security is often achieved using SSH (e.g. openssh) which secures the network connections in a attractive way.

### 0.2.3 *RT*

RT is used by openssl as a incident tracker.

### 0.2.4 *Doxygen*

This is tool to generate html documentation directly from source trees.

### 0.2.5 *Mailing lists*

Mailing lists are used in different forms to inform interested parties and to contact developers. There exist a great variety of mailing list tools, themselves available as open source.

### 0.2.6 *HTTP/FTP distribution*

Snapshots and releases of software are available using standard http and ftp servers. The http services are also used for all relevant documentation distribution.

### 0.2.7 *Gnu autoconf*

The GNU autoconf tools provide an efficient mechanism to ensure portability of software. If these tools are used from the very beginning of the project, they are quite efficient. Conversion to them after some initial integration is a rather complex process.

### 0.2.8 *Coding conventions*

Coding conventions are at least a useful means to enhance the quality of the software. In addition, they are required to enhance the portability of the code, in particular between Windows and Unix platforms. The rather old conventions and tools to ensure portability between K&R C and ANSI C is no longer a real issue. Coding conventions are required in common criteria.

## 0 Annex D: Collaboration environment overview

Project management and information exchange will need different tools and usage procedures. Among them are collaboration servers whose requirements are:

- Must allow the realisation of project tasks by his responsible (role definition and access control),
- There must not be any need of any intermediary for these realisations, neither for communications,
- System administration and application management can be de-correlated,
- Advanced security for system, connections, data.

Note: Details for setting up the collaborative server requirements are bound to the acceptance of EAL work context.

The following collaborative servers are identified (proposal):

- The Public Web Site
- The Public FTP Server
- The Public News and/or mailing list Server
- The Private Web Site
- The Private Source Repository
- The Public Source Repository

### 0.1 Public Web Site

On the public Web site ([www.eupki.org](http://www.eupki.org)) the following services are set-up:

- Bug report submission form,
- Enhancement/Feature request form,
- Beta and Release versions download page,
- User Documentation download page,
- Beta and Release versions download management (allow management of the Beta and Release versions download page by integrators)
- User Documentation download management (allow management of the User Documentation versions download page by integrators)

### 0.2 The Public FTP Server

On the public FTP server ([ftp.eupki.org](ftp://ftp.eupki.org)) the following services are set-up:

- Beta and Release package download (anonymous access).
- Patch download (integrators).
- Beta and Release package upload (integrators).
- Patch upload (integrators).

### 0.3 The Public mailing list Server

This server allows exchange of ideas between all project actors. These activities are organised in groups. Groups can be set up on an externally hosted server.

We propose to define the following groups:

- eupki-developer : development and specification matters
- eupki-user : user oriented discussions (new features, usage, installation, ...)
- eupki-bugs : error reports.

All mailing lists should be archived, and the archive should be accessible using some tools like hypermail. A simple textual search mechanism should be included.

## 0.4 Private Web Site

On the private Web site ([extranet.eupki.org](http://extranet.eupki.org)) the following collaborative tools are installed:

- Bug report submission form (high priority),
- Enhancement/Feature request form (high priority),
- alpha versions download page (including Integration and tests reports),
- alpha versions download management (allow management of the alpha versions download page by integrators)
- User Documentation download management (allow management of the User Documentation versions download page by integrators)

Only consortium members have access to this Web site.

Bug reports and feature requests are only managed by mail for now. If it becomes hard to handle we will consider other solutions (rt, gnats, bugzilla).

## 0.5 Private Source Repository

This is the internal source repository. It is managed with CVS. WP5 members are in charge of its administration and regular data backup.

Only Developers and integrators have read/write access to it. All changes are reported by mails to developers and integrators. Other Consortium members may have read access and may receive commit e-mails upon request.

Secure access to this repository must be set-up.

## 0.6 Public Source Repository

The Public source repository is the exact replication of the Private Source Repository. Its main goal is to give access to the project source code to external contributors whilst preventing the main Source Repository from being corrupted.

There is only a read access to the repository through CVSWeb. Commits are not permitted. Anybody may export the repository by using a CVS client.

It is updated once a day.