



Distribution List :	Project Partners
Author :	Lionel Auroux, Paul-André PAYS, Peter Sylvester Groupe ON-X Sébastien Kuhn EADS
Distribution List :	Project Partners
Authorised by :	Paul-André PAYS
Date of Issue :	April 30 th , 2004
Issue :	1.0
File Name :	EuropePKI-WP6-D6.3-V1.0.doc
Work Package :	WP6 Integration
Deliverable Number :	D 6.3
Deliverable Type :	final
Deliverable Nature :	Integration report
Total Number of Pages :	24
Contact Details for EUPKI :	Project Coordinator Alain ROUX GIP-MDS mail : alain.roux@gip-mds.fr web site : www.europepki.org

Table Of Contents

TABLE OF CONTENTS.....	2
0 DOCUMENT CONTROL	4
0.1 ABSTRACT	4
0.2 KEYWORDS.....	4
1 MANAGEMENT OVERVIEW	5
1.1 EXECUTIVE SUMMARY	5
1.2 SCOPE STATEMENT	5
2 INTEGRATION CONTEXT OVERVIEW	6
2.1 WP6 ORGANIZATION AND TASK SPLITTING.....	6
2.2 RELATIONS BETWEEN WP6 AND WP5	6
2.3 PLANNED AND ACTUAL SCHEDULE	6
2.4 RELATIONS BETWEEN WP6 AND THE OTHER EUPKI ENTITIES	7
3 PRESENTATION OF THE EUPKI SOFTWARE	9
4 INTEGRATION ACTIVITIES REPORT	10
4.1 INITIAL ACCEPTANCE OF INDIVIDUAL MODULES.....	10
4.2 ACTUAL INTEGRATION ACTIVITIES	10
4.3 STATISTICS	11
5 MAIN DIFFICULTIES ENCOUNTERED AND SOLUTIONS PROVIDED	12
5.1 DELAYED AND CHANGED MODULE DELIVERY SCHEDULE	12
5.2 DEVELOPMENT DOCUMENTATIONS	12
5.3 SOFTWARE QUALITY	13
5.4 ISSUES AND SOLUTIONS	13
5.4.1 <i>Temporary and unused code</i>	13
5.4.2 <i>Inter-module communication</i>	14
5.4.3 <i>Key ceremony and PKI initialization</i>	14
5.4.4 <i>X509 DN encoding</i>	14
5.4.5 <i>Basic constraints encoding</i>	15
5.4.6 <i>Certificate version encoding</i>	15
5.4.7 <i>Certificate serial number encoding</i>	15
5.4.8 <i>Security officers role access control</i>	16
5.4.9 <i>Certificate Request extensions</i>	16
5.4.10 <i>PKCS12 retrieval</i>	16
5.4.11 <i>Multi-CA support</i>	17
5.4.12 <i>Authentication applet complexity</i>	17
5.4.13 <i>RA profile management</i>	17
5.4.14 <i>Session Authentication</i>	18
5.4.15 <i>Jboss deployment</i>	18
6 ACHIEVEMENTS	19
6.1 CVS 2 CONTENT SUMMARY	19
6.2 READY TO INSTALL PACKAGE	19
6.3 INSTALLATION AND USAGE DOCUMENTATION	19
7 ASSESSMENT OF THE RESULTS OBTAINED.....	20
7.1 PRINCIPLE GOALS	20
7.2 WP7 FEED BACK TO WP6	20
7.3 WP6 FEED BACK TO WP5	21
7.4 POTENTIAL USERS' COMMUNITY ACCEPTANCE GUESS.....	21
7.5 OPEN SOURCE DEVELOPERS' COMMUNITY ACCEPTANCE GUIDE.....	21
0 ANNEX A : COLLABORATION PROCESS.....	22

0.1 OVERVIEW 22
0.2 ACTORS 23
0 ANNEX B: INTEGRATION GLOBAL PROCESS PRINCIPLES..... 24

0 Document Control

<i>Issue</i>	<i>Date of Issue</i>	<i>Comments</i>
0.1	January 2004	Creation, internal working version ON-X
0.2	February 2004	Initial version containing global canvass reviewed by ON-X and EADS.
0.3	February 2004	Initial problem list
0.4	February 2004	Updated problem list
0.5	March 2004	Updates after feedback from WP7 and corrections
0.6	March 2004	Initial summaries
0.7	March 2004	Final cvs organisation
0.8	March 2004	Final internal draft
1.0	April 2004	Final version

0.1 Abstract

The purpose of the deliverable D6.3 is to provide the final report about the actual software integration activities and to present the achievements of WP6.

0.2 Keywords

EUPKI EUPKI, the libre software Public Key Infrastructure (project name)
EuropePKI The software provided from the EUPKI project
WP6 Work Package 6

1 Management Overview

1.1 Executive Summary

This document presents the integration activities performed within WP6 for the EUPKI project.

The main sections are :

- Integration activities report
- Main difficulties
- Solutions provided or proposed
- Achievements

The document also summarizes flows and tools used for the whole integration of the EuropePKI software.

We note that the initially planned deliverable D6.2 is void, since no preliminary version of the software had been produced. The complete integration reporting is thus in this D6.3 document.

1.2 Scope Statement

The scope of this document is to report about the different activities directly related to the integration of the software and to make a short self-assessment of the achievements.

Reference	Document
R6.1	Integration Process Overview
R6.2	Integration guidelines and acceptance procedures
R6.4	Scenario proposition
R6.5	EuropePKI installation and usage Guide

2 Integration context overview

2.1 WP6 organization and task splitting

ON-X had the leadership of the integration work. The work load was shared between ON-X and EADS in the following way. Actual first integration was done by ON-X, testing and several documentation was performed by EADS. EADS served as an alpha-tester of the software.

2.2 Relations between WP6 and WP5

The teams of WP5 and WP6 cooperate according an integration process that is described in detail in the document R6.1 (Integration Process Overview). Software is delivered by WP5 according to the rules described in document R6.2 (Integration Guidelines and Acceptance Procedures).

For different reasons, the technical committee had decided to use three different cvs systems in order to coordinate the work between the development and integration teams. These systems are:

- CVS0: an internal system used by WP5
- CVS1: a system created from the WP5 module deliveries. All source code is slightly modified to reflect the agreed copyright notices.
- CVS2: This is the system used by WP6 for open publication of the integrated system.

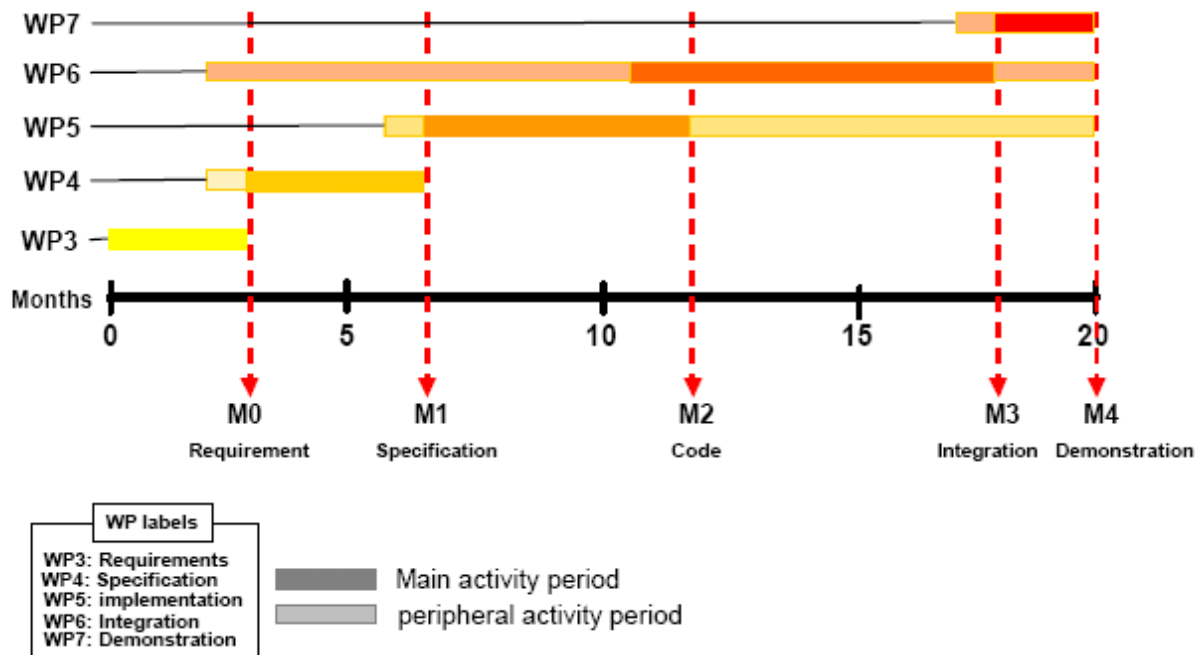
It has been suggested by the evaluators that three different systems are not a very efficient way to handle the project. This is in fact true. At least the CVS1 has technically no longer a sense after the last delivery of WP5; it is necessary that either the CVS2 will be the base for future developments, or that the contents get resynchronized into the CVS0. The small number of enhancement that appeared in CVS0 two days after the last can be easily integrated..

2.3 Planned and actual schedule

There are at least three different schedules concerning the WP6 activities.

- The DOW contains an estimation indicating already some large overlap between WP5 and WP6 activities (cf. figure below)
- The overlap has been furthermore enhanced by different phases of the software. (cf. annex B)
- At last, there were several versions of the module list to be delivered in a total of 8 deliveries.

The following figure shows the initial schedule of the DOW.



The following figure shows the proposed schedule of work after the amendment.



The actual work in WP6 lasted up to the end of the project in order to react to WP7 feedback.

2.4 Relations between WP6 and the other EUPKI entities

The initial work of WP6 consisted in participation and review of WP3 and WP4. Furthermore, cooperation with WP8 took place at several moments, in the beginning in order to validate and discuss the integration procedures etc, and after the main WP5 activities to benefit from the code inspection work done by WP8.

All WP6 proposals were systematically presented to and discussed with the technical boards. Decisions, whenever needed, were made by the Steering Board.

3 Presentation of the EUPKI software

The EuropePKI software consists of a distributed system of three main components:

- A registration authority component managing a data base of certificate holders in a hierarchical company structure, allowing to request key generation and certification.
- A certification authority component allowing to perform certification.
- A central key generation service.

All services communicate with an SQL database maintaining the state of the PKI.

The services are developed in **java** running in **Jboss** application server environments.

Bouncycastle is used as cryptographic service provider.

Operators communicate with the server using a navigator. The software comprises an applet used for authentication and for signing of certification requests.

The software allows creation of certificates signed by one of several CA keys. CA keys are generated during a particular key ceremony.

Operators can control the various functions of the components. The access to functions can be controlled through roles.

4 Integration activities report

4.1 Initial Acceptance of individual modules

The first elements checked and which required a decision by the Steering Board were :

- Documentation
- License Boiler plate

A solution accepted by all partners was eventually found and has been implemented by all partners.

4.2 Actual integration activities

The actual integration work consisted of several major parts:

- Code review and testing of individual modules including reverse engineering of specifications
- Integration of modules and development of missing parts
- Preparation of an installable and usable system
- Provision of an installation and usage guide.

After the first deliveries of code, the major activity consisted of preparing a tool to create an installable system from the delivered source code, i.e., to repackage the source into deployable jboss configurations. Since it was clear that in small environments the software needed to be used on single machine, it was necessary to find an appropriate structure.

The first attempt was to have one JBOSS environment and the three different applications deployed.

This did not work because of some naming conflicts in the EUROPKI modules, thus it was decided to have three different JBOSS environments, each of them having all non-default parameters for ports etc. Thus, it is possible to either run them on one machine, or on different machines in the same way.

Since it was obvious that the integration needed to take into account several code deliveries, WP6 developed a script to create the installable package from the development sources.

The installable package contains the complete JBOSS environments.

The logical architecture of the provided system is summarized as:

- Each of the three EuropePKI components (RA, CA, KGS) runs on a logically independent machine whose network names are fixed to be {RA,CA,KGS}.europepki.lan. Depending on the actual network structure, hostnames and/or DNS must be configured accordingly.
- Each component uses different unix logins and different non-default jboss configurations, thus, they can all run on one machine without conflict
- Each component communicates with a mysql server that must be locally available on the machine where the component is installed.

- The CA component publishes CRLs and certificates towards an external LDAP directory which must be reachable with a hostname ldap.europki.lan

The package generation and installation scripts were developed long before the final deliveries and turned out to be effective.

The bottom up nature of the delivered software required to develop or to redevelop missing or inappropriate or incorrect parts of the software. It was possible to complete the software to have a working system.

A usage guideline was planned and developed to document the tested ways of using the software.

4.3 Statistics

It is somewhat difficult to correctly number the amount of code changed or added during WP6 activities. A recursive diff between the last delivery and the modified code shows around 10000 lines of differences not counting the additional packaging and installation scripts.

5 Main difficulties encountered and solutions provided

In this chapter we summarize the problems encountered during WP6 activities and describe the solutions that have either been implemented or proposed as enhancements. Blocking problems have been fixed. Four different categories are treated..

5.1 Delayed and changed module delivery schedule

The schedule of the modules deliveries had been changed several times.

Furthermore, the order of modules to be delivered had been changed considerably; This latter point had important consequences. The initial schedules provided for a rather top-down (or at least yoyo mode) oriented delivery of modules which would have allowed to create an integrated version very early containing hard coded dummy modules, and also to validate the provided functionalities against the specifications and user requirements.

Due to the changes coming from WP5, the real integration was considerably delayed, and needed to be compensated by an extension of the project life time and some restrictions concerning the integration work. In particular, it was no longer possible to deliver a preliminary integrated version, as there were no "preliminary development version".

These changes accepted by the Steering Board, had not only impact on WP6 but also on the "customers" of WP6, mostly WP7.

5.2 Development documentations

An important problem was the lack of any development design documentation. Thus, in order to understand how the specifications had been translated into actual code, it became necessary to read the source code itself. Furthermore, the bottom up schedule of the development did not facilitate the work, There is a large amount of unused code in the CVS0/CVS1 used for deliveries from WP5 and WP6. Combined to the lack of documentation it was difficult to distinguish unused code from actual "living source code" until the final delivery during December 2003. The dead code not debugged added an additional amount of confusion.

The module functions are formally documented using javadoc, this is important and necessary.

The experience confirmed, (easy to guess) that this is in general not very helpful when low level rather formal code documentation is not complemented by some higher level development documentation presenting the global implementation design and architecture. (This is by no means unique to this project, even long exiting project like OpenSSL still partially suffer from that problem).

The individual source components have no documentation of intended input/output. This revealed as a source of complexity for the integration work, as the work as to rely on a pragmatic combination of i) guesses and ii) trials.

5.3 Software quality

The structure of the source tree provided by WP5 is such that in many cases the same code is used at least three different places, since the three main components (RA, CA, KGS) don't always share common elements. The consequences were :

- some name clashes and,
- supposedly identical modules which were not corrected in the same way in all places.

Errors related to these problems have been corrected, but this task revealed to be really complex and time consuming.

Due to the bottom up approach, there is a lot of encapsulation and glue code that is never used at all in higher layers, thus, not tested.

Although some appraisable attempts have been made by WP5 to create abstraction for basic types, the actual implementation had not taken full advantage at all of the possibilities of java in this domain. Though identified, this type of not blocking problems has not been corrected by WP6; this could only have been considered as feasible for a second major release, which the tight schedule did not allowed.

Exception handling is often processed in a very inconsistent way making it often difficult to detect the real problem due to re-raise of a new exception. This aspect was also included in the list of "nice to have" improvements for a second major release, but for practical reasons it was not realistic to re-thing and redevelop "exception handling" and thus, WP6 work was limited to a case-by-case bug fixing and improvement.

There were many debugging or fixing code pieces left in the code. In the best case, they were marked with a "FIX ME" comment; WP6 had to guess whether the fix had already been done or was still to be conceived, designed or implemented.

5.4 Issues and solutions

In this subchapter we only describe in detail the few most important problems discovered during integration, and the solutions, which were either implemented or suggested to WP5.

The minor code changes performed by WP6 are far too numerous and thus, are not documented here. They are all reflected in the latest version available on CVS2.

5.4.1 *Temporary and unused code*

Due to the bottom up nature of the development, and to only some flash integration performed by the developers, there was a lot of code containing hard coded values for various elements that need to be configurable. Furthermore, there was code that

initialized the database content during a very first operation of a component. All this code needed to be removed, deactivated and replaced by a configurable mechanism.

On the other hand, a large amount of code was not used at all, and of course, this code contained lot of errors, which were detected during code inspection before it was possible to understand that the code was not used at all resulting in an important amount of loss of time.

There were also some places where objects like pkcs12 files were stored in a file as part of some debugging activity.

5.4.2 *Inter-module communication*

WP5 had delivered a jmx queue management tool that uses a particular code to circumvent some bug in jboss.

The problem consists of distinguishing whether a remote queue jndi service is either provided by the local jboss or by another jboss instance and to invoke them differently.

The logic used to determine whether system is the local one assumed that a remote jboss actually runs on a different machine and does not provide for having more than one jboss on the local machine.

It took a rather considerable amount of time to detect the origin of the problem. It seems that the original code was not developed directly by WP5, since there are many code features that are not used or required by the EuropePKI software.

The code was corrected in order to permit the EuropePKI modules to communicate.

5.4.3 *Key ceremony and PKI initialization*

The delivered key ceremony modules had several defaults.

First, they didn't work at all, it seems that some very last minute code changes had been made without any regression tests.

In a first phase, some obvious bugs had been corrected by WP6 allowing creating some CA certificates and root administrator certificates.

In order to simplify the PKI initialization, the key ceremony –as fixed by WP6- not only creates PKCS12 containers for keys and certificates but also a corresponding X509 certificate. This is used for example to initialize the master operator in the servers.

As delivered by WP5, the key ceremony and some module initialization use several different configuration property files that would require a lot of configuration work.

In order to simplify the PKI installation, the key ceremony CA configuration have been concentrated by WP6 in one “properties” file per CA, containing all necessary parameters to allow an automatic initialization of the CA server after a key ceremony.

In order to initialize the servers after the key ceremony, in particular the CA and RA, startup and initialization modules have been developed by WP6.

5.4.4 *X509 DN encoding*

Bouncycastle provides different methods to manipulate DNs.

The WP5 developers have chosen almost systematically the simplest interface, which were unfortunately not appropriate to create correct DN encodings. *Bouncycastle* provides an API to handle DNs as a unordered hashtable (with OIDs as keys).

This is an acceptable feature for a directory search since it is more than uncommon to have two DN which only differ in the order of the RDNs, but this API does not permit to control the ordering of the encoding.

All pieces of source code having to encode DN needed to be corrected by WP6 to use one of the other APIs of *bouncycastle* enabling a legal and conformant encoding of the DN within the certificates.

5.4.5 *Basic constraints encoding*

During the integration of the key ceremony modules it was discovered that the *Bouncycastle* implementation of a “basicconstraints extension” was totally broken: the standard *Bouncycastle* implementation does not encode correctly the extension, and furthermore, breaks during decoding a correct encoding.

A new module was developed, since one cannot replace directly parts of *Bouncycastle*.

This module was contributed to the *Bouncycastle* development community for inclusion. It is available since April 10th 2004 in version 123:

Excerpt from the contributors.html:

Peter Sylvester <Peter.Sylvester@edelweb.fr> - improvements to the ASN.1 BasicConstraints object.

5.4.6 *Certificate version encoding*

Although there was an attempt to create constant for the X509 certificate versions, there was some place where the “*setversion*” method was called directly with an integer 3, which would have given an invalid version.

This error demonstrates two problems. Either, a subtype of integer should have been created and only values of that type should have been allowed as parameter. Or, alternatively, a *setversion3* method should have been defined.

In fact, the whole API of setting different versions is totally superfluous; since all generated certificates have a fixed version anyway.

In other words, the developers have wasted time by coding a totally superfluous API.

This error was detected by WP8 through code inspection.

5.4.7 *Certificate serial number encoding*

The code to create unique serial number is mainly based on a millisecond counter since epoch time. It seems, that in order to have smaller (or larger) serial numbers, there is an additional number defined in a properties files for each CA which is added to this value. The maintenance of this parameter is error prone, non-unique serial numbers may be the result.

WP6 removed the usage of this parameter; all serial numbers are now simply milliseconds since epoch start.

5.4.8 Security officers role access control

Conceptually, all activities of operators are controlled by an access list of a role. According to the rights, any operator only sees the available functions for its specific role in its interface.

Unfortunately, an unclear behavior of Jboss creates an java exception.

The error was known by the developers and circumvented by simply showing all possible operations.

There are several possibilities for a solution, which all more or less directly use an sql access function to the "roleservices table".

Due to time constraints, WP6 did not develop the necessary code for the current version.

5.4.9 Certificate Request extensions

A small number of private extensions in certificate requests are used by the RA and CA components; extensions are defined by OIDs.

WP5 used a part of the standard extensions branch. This **unauthorized** use was replaced by WP6 by an oid arc defined under the official and registered EdelWeb company arc: **1.3.6.1.4.1.5309.3.1.1**

5.4.10 PKCS12 retrieval

One of the desired operational scenarios consists to centrally generate keys for end users and to export them via a pkcs12 files.

In order to do this, the core part of the transaction processing of the CA needed to be modified by WP6, and an additional extension concerning the provision of a protection password by the RA was also developed.

A user provides a password during online registration at the very end of the registration process. This password is transported in the certificate request as a private extension.

Due to time constraints and the necessity to make a package available to WP7, the password is currently transported in clear. It should be protected for example by encrypting it using the CA public key. *This is left for future development.*

The transaction process of the CA (the message driven "*beans*") needed to be changed by WP6 in order to communicate the password to the KGS, and to obtain the final PKCS12. The original code sent the created certificate back to the RA before creating the final PKCS12 stored in the KGS.

Some additional changes by WP6 in the RA CA protocol format were necessary to transport the PKCS12, and in the RA, the PKCS12 is now represented as a private extension of the initial request.

In order to download the PKCS12 and also a DER certificate, a new module had been developed by WP6.

5.4.11 Multi-CA support

The software provided by WP5 only supported one single CA. During the modification of the transaction system, support for multiple CAs was added by WP6. This modification also allows a better separation of operator access control and operational CAs.

It is assumed that there exists always one “service internal” CA used only to defined operators and to protect the EuropePKI components.

5.4.12 Authentication applet complexity

The authentication applet is currently not a real end user oriented tool.

- The features provided are too technical;
- Some features are not implemented, e.g. PKCS11 support.

The interface to PKCS12 files distinguished the two potentially different passwords for integrity and key protection. This was simplified by WP6; the two passwords must now be identical. The known risk concerning a brute force attack to the MAC, and the workaround using a high iteration count has not been addressed.

In order to eliminate one configuration parameter, a change was made to use always the first key available in the PKCS12 file. While developing this modification, it was discovered that the “*friendlyname*” attribute has a very restricted character set, for example no minus sign can be used in such a name. (Totally unrelated: Netscape/Mozilla does not like colons.)

The most important point is the requirement for a local privilege configuration, which is rather difficult to handle. A solution consists of having the applet signed (e.g. by the EuropePKI administration authority). This was suggested as an improvement to WP5 by WP6. The estimated work load is two days.

5.4.13 RA profile management

The delivered code from WP5 concerning registration profiles was totally irrelevant. Some logic of profile checking in the CA components was simply copied into the RA modules. This was useless, since it only consists of defining which certificate extensions are created in the certificate requests. It became necessary for WP6 to largely change the way how RA profiles are used.

Furthermore, the existing registration forms had a *hard coded* mapping of registration information from company/department/holder into DN components (including the previously described problem of DN ordering).

The newly developed code by WP6 consists of providing an interface to use jsps as implementation of a profile. The jsp provides a form interface that will generate all necessary information to create the certificate request. The profile jsps have the

responsibility to map information from the registration into a certificate request including extensions.

Several profiles for typical end entities like ssl users and server and email users have been developed and are provided as example in the source code.

This solution is definitely not optimal since it required development of code and its installation. There are several possible ways to enhance this, which require development of additional code in the RA components.

During the described developments it was discovered that (again due a usage of a hashtable), it is not possible to generate more than one certificate extension with the same OID, e.g. subject alternate name extension.

5.4.14 Session Authentication

The authentication and signature applets communicates with three servlets (Authenticate.java). Strangely enough the code exists three times and was not folded into a common component. The code had to treat an HTTP POST request but didn't manage to read more than one TCP packet.

WP6 corrected the error.

5.4.15 Jboss deployment

The Jboss deployment (startup of components) can be considered as a real disaster. It is not really possible to control dependencies between components, i.e., ensure that they are deployed in the right order.

There exist some undocumented features that are discussed in mailing list describing how the order of deployments can be influenced.

Nevertheless, the order depends in some sense on dates of the generated files with some obscure bugs.

Note: this is obviously a JBOSS problem and not a developer's problem.

WP6 finally found a solution that allow to correctly deploys all the components without spurious errors.

JBOSS components are often deployed in the form of archives. Since each of the EuropePKI components consisted of a few archives, it was difficult to modify and test on the fly the operation of jsps for example to add profiles.

The RA software structure by WP6 was modified to allow online modification with redeployment.

6 Achievements

6.1 CVS 2 content summary

The CVS2 contains all the basic source code including installable package generator. During the integration work, the installable package including all required components was also provided in the cvs.

This was useful for WP7 but technically very problematic due to how binary versions are handled by cvs.

The installable package of the most recent version contains nearly 55 Megaoctets of compressed data.

6.2 Ready to install package

A script to create an installable package from the source code was developed by WP6.

The package regroups the various source modules into jboss deployable components, and created the three different JBOSS environments.

The results can be used to install the three different components on one or more machines.

A configuration file is used to parameterize the installation process..

6.3 Installation and usage documentation

A document describing an installation procedure of the whole software and some usage documentation has been developed. This is available as document R6.5

Since some portions of the software have not been completely tested, there unfortunately exist some undocumented features.

The current documentation includes :

- Installation of all prerequisites
- Installation of EuropePKI components
- Configuration for key ceremonies
- Configuration for clients access
- Initialization of the PKI
- Operation of the RA and CA components using examples.
- Registration profile principles

In order to understand the actual software, it is also necessary to be familiar with the WP4 specifications (see deliverables D4.x), in particular, the data base content.

7 Assessment of the results obtained

In this chapter WP6 propose a self assessment of its results.

7.1 Principle goals

The main tasks of WP6 consisted in:

- taking the results of WP5,
- making a working system out of it,
- providing useful feedback to the developers,
- delivering the software to WP7 according to the agreed scenarios,
- assisting WP7 in the installation, and
- obtaining and reacting to feedback WP7.

Additional inputs came from WP4 and WP8.

The schedule of delivered software was changed considerably, implying that a best effort and economic use of time and manpower had to be made to recover from the timing problems.

7.2 WP7 feed back to WP6

The user community is represented, within the project, by WP7.

There was only a very short time deliver a “workable” package enabling them to perform actual tests.

During the initial installations (already with EADS as internal alpha-tester) it turned out that the developed software needed a lot of additional effort to minimize configuration problems.

The major problems seen were difficulties in the local client java configuration, distribution of the operator credentials, and correct sequence of key ceremonies and initialization.

Parts of these problems have been addressed by simplifying the properties configuration and the PKI server initialization procedures.

It is obvious that additional work remains for the development, in particular to minimize the use or get rid of specialized properties files.

The simple operational scenarios developed by WP6 and proposed to the user community permitted to create the intended certificated.

7.3 WP6 feed back to WP5

Several comments had been returned by WP6 to the WP5 developers.

Since the schedule of integration was also very tight before project end, nearly none of the comments had been transformed into actual code changes by WP5 before the end of the project.

Furthermore, the WP5 developers were not easily available during the main “debugging” activity phase of WP6, i.e. after the final WP5 delivery in December 2003.

WP6 therefore implemented corrections and additional features directly in the code available on the project CVS.

WP6 duly informed WP5 in particular during the technical board meetings during which important changes proposed were announced before being implemented.

All propositions have been accepted as useful features.

We are rather confident that the future developers will implement the additional enhancements that have been presented but not implemented by WP6.

7.4 Potential users' community acceptance guess

It is difficult to estimate the interest of potential users of the EuropePKI software, mainly because of the still immature status. Nevertheless, the following list of points seem to us important motivation aspects:

- The software is totally open source
- A distributed architecture is the base and not a simply add on.
- The code is written in an architecture independent language (java).
- The software integrated a flexible privilege management for operators based on roles.

The implementation of necessary enhancements seems possible in small steps, enhancing the usability to more and more users. The challenge is to motivate end users to formulate requirements for operational scenarios and get them implemented in cooperation with some development team.

7.5 Open Source developers' community acceptance guide

If one compares the situation with other comparable open source projects like OpenSSL, OpenCA, newPKI, newClear, etc; one can clearly identify that the number of active developers is very small, and there are also a rather small number of active contributors.

The actual situation after the project end is that there are no active developers for EuropePKI. Axetel has announced that a reasonable amount of developers will be made available which can lead to a comparable developers' support.

It will be necessary to clean up the architecture in order to enhance the security among components, to address problems of performance, and to add sufficient flexibility allowing a fast integration in real scenarios while at the same time avoiding to create a overly complex system with too many parameters which seems to be for example a tendency with some software based on OpenSSL code.

0 Annex A : Collaboration process

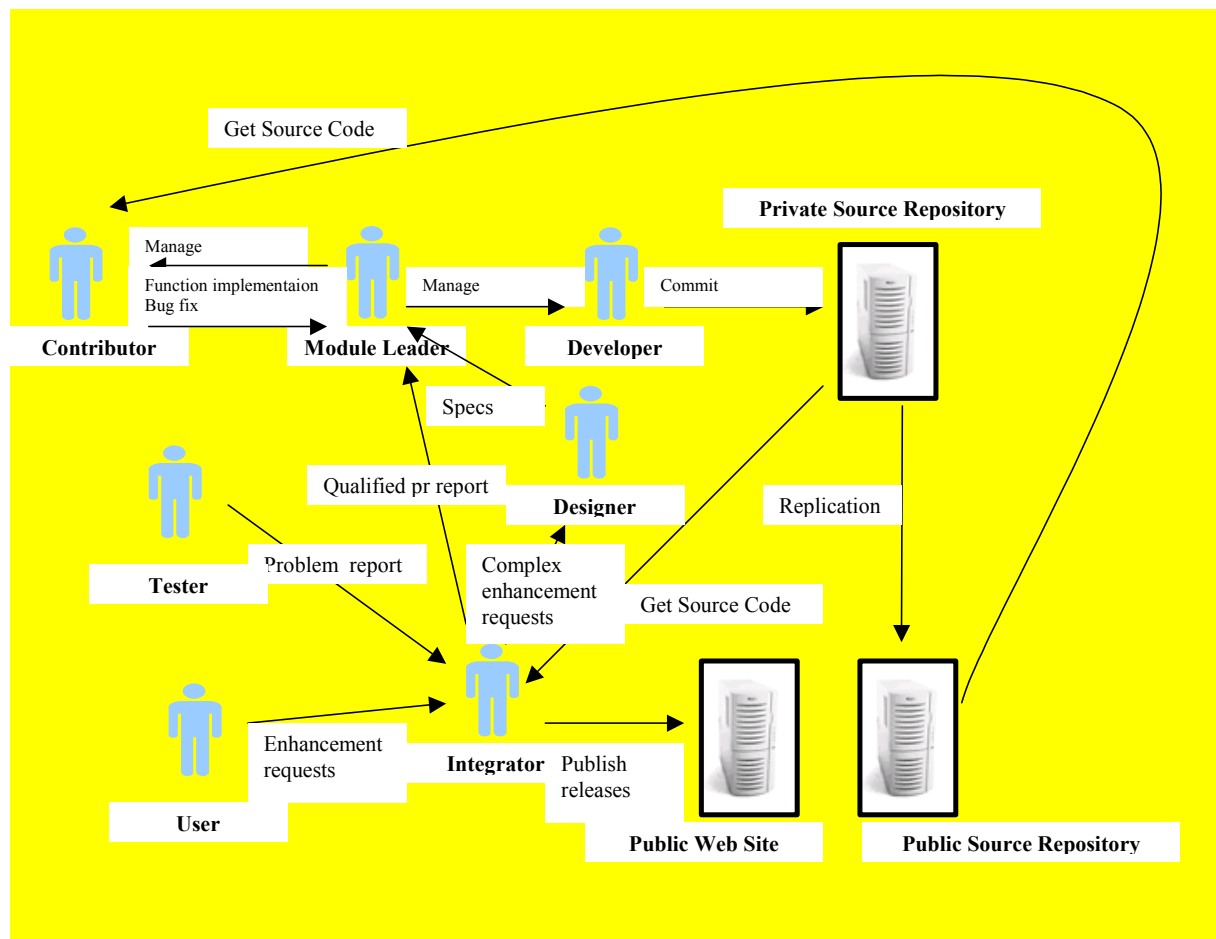
This annex summarizes the global “Integration Process Overview”, document to be produced.

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 **Erreur ! Source du renvoi introuvable.**
“Erreur ! Source du renvoi introuvable..

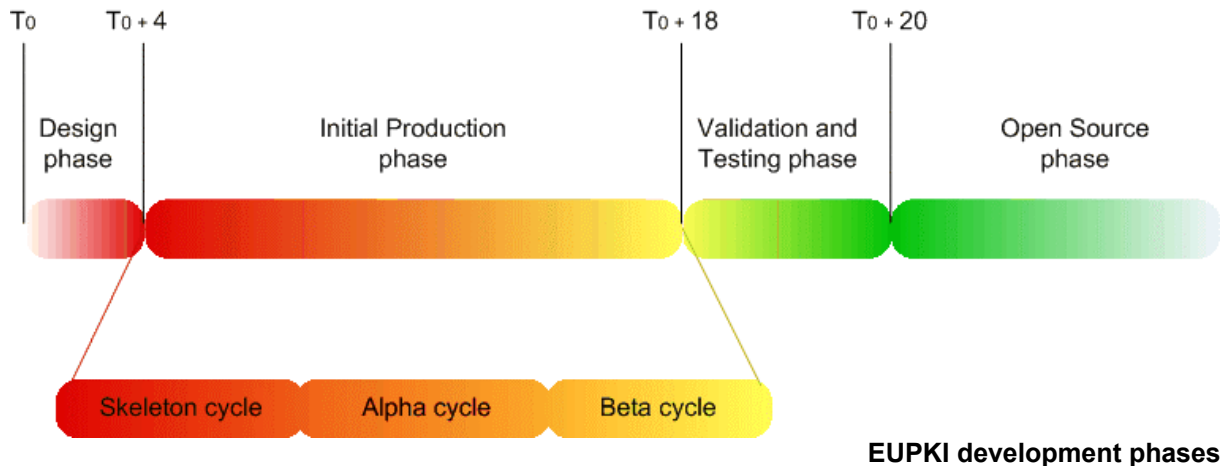
0.2 Actors

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"