OpenCA Guide for Versions 0.9.2+

Michael Bell <michael.bell@web.de>
Chris Covell <chris@katjam.co.uk>

OpenCA Guide for Versions 0.9.2+

by Michael Bell and Chris Covell

Table of Contents

Introduction xii
I. Design Guide 1
Preface iii
1. General Design 4
1. Basic Hierarchy 4
2. Interfaces 5
2.1. Node 6
2.2. CA 6
2.3. RA 6
2.4. LDAP 6
2.5. Pub 6
3. Configuration 7
4. Database 7
5. Interface 7
6. Lifecycle of the objects 7
7. Sub-Ca 8
7.1. Example 1 8
7.2. Example 2 8
2. Recommendations 9
1. Hardware Issues 9
1.1. Time 9
1.2. Failing disks 9
1.3. Hardwaremonitoring 9
2. Physical Security 10
2.1. Safes and Dataorganization 10
2.2. Buildings 10
3. Network Issues 10
4. Certificate Issues 11
4.1. CDPs 11
4.2. Application specific problems 12
5. Organizational Aspects 12
5.1. Dual Access Control 12
5.2. Privacy vs. Security 12
5.3. Enforcement of Access Control 12
5.4. Privacy Officer Integration 12
5.5. Enterprise Integration 13
5.6. Parallel use of several end user PKIs 13
II. Installation and Configuration Guide 14
Preface xvii
3. Installation 18
1. Preparations 18
1.1. Software 18
1.2. Hardware 18
2. Configure 18
2.1. Host System Configuration 19
2.2. Filesystem paths 19
2.3. Webserver specific stuff 20
2.4. Email 20
2.5. Compiling features 20
3. Installation 21
4. config.xml (for RPMs and DEBs too) 21

4.1. Configuration sections of config.xml 22
4.2. How to setup two management interfaces on one server? 24
4. Configuration 26
1. Access Control 26
1.1. Channel verification 27
1.2. Login 27
1.3. Session management 29
1.4. ACLs 30
2. Token and keyconfiguration 32
2.1. OpenSSL 33
2.2. Empty 34
2.3. LunaCA3 34
3. OpenSSL 34
3.1. Certificate Extensions 34
3.2. Profiles 36
4. CSRs 38
4.1. Additional Attributes 38
4.2. PKCS#10 Requests 39
4.3. Basic CSR 39
4.4. SCEP 41
5. Subject 41
5.1. Common stuff 41
5.2. dc style 42
6. Subject Alternative Name 43
7. LDAP 44
7.1. Configuration of the Directory 44
7.2. Configuration of the online components 44
7.3. Writing Certifciates to the Directory 44
8. SCEP 45
8.1. OPENCADIR/etc/servers/scep.conf 45
8.2. OPENCADIR/etc/config.xml 45
9. Dataexchange 45
9.1. Configuration 45
9.2. Adding a new node 49
10. Databases 49
10.1. PostgreSQL 49
10.2. DBM Files 51
11. Email 52
11.1. Sendmail with basic SMTP authentication 52
III. User Guide 55
Preface lviii
5. Interface Descriptions 59
1. Public PKI Server 59
1.1. CA 59
1.2. User 59
1.3. Certificates 61
1.4. Requests 62
2. Registration Authority 63
2.1. Administration 63
2.2. Pending Requests 63
2.3. Information 64
2.4. Utilites 64
3. Registration Authority Node 64
3.1. Gateways 64

3.2. Administration 65
3.3. Utilites 67
4. LDAP Interface 67
4.1. Update LDAP 67
4.2. View CA-Certificates 68
4.3. View Certificates 68
4.4. View CRLs 70
6. Functionality Descriptions 71
1. CA Initialization 71
1.1. Phase I: Initialize the Certification Authority 71
1.2. Phase II and III: Create the initial administrator and RA certificate 73
2. Node Initialization 75
3. CSR Handling - a request HOWTO 76
3.1. Ways to request a certificate 76
3.2. Edit a certificate signing requests 78
3.3. Approve certificate signing requests 79
3.4. Issue a certificate from a certificate signing request 79
3.5. Certificate enrollment 79
3.6. Delete certificate signing requests 79
4. Certificate Handling 79
4.1. Find a certificate 80
4.2. Download 80
4.3. Start revocation 81
4.4. Write an email to the owner 81
4.5. Informational messages and their meaning 81
5. SCEP 81
5.1. SSCEP 81
5.2. NetScreen ScreenOS 83
5.3. F-Secure VPN+ 83
7. Client Support 85
1. Introduction 85
2. Mozilla 85
2.1. General 86
2.2. Mozilla 87
2.3. Netscape 4 87
2.4. Opera 87
3. Microsoft 87
3.1. Internet Explorer 87
3.2. Outlook 87
3.3. Outlook Express 87
IV. Technology Guide 88
Preface xci
8. Introduction 92
1. Slotechnology 92
9. XML 94
10. Cryptolayer 95
11. Accesscontrol 96
12. Logging 101
13. Webinterfaces 103
1. Interfacebuilding 103
1.1. Technology overview 103
1.2. Customization capabilities 105
2. CSS 106
3. Configuration after installation 106

14. Hierarchy 107
1. Nodemanagement 107
2. Dataexchange 107
15. LDAP 108
1. Supported options in interface configurations 108
2. LDAP schema specification 109
2.1. Used objectclasses 109
2.2. Supported attributes 110
2.3. Common definitions for distinguished names 110
2.4. Special definitions for user certificates 111
3. Sourcecodeorganization 112
3.1. Structure of the code 112
3.2. The relevant commands 112
3.3. export-import.lib 113
3.4. ldap-utils.lib 113
3.5. OpenCA::LDAP 113
16. Batch System 114
1. Requirements 114
2. Design 114
3. Data Import 115
4. Change the workflow 117
5. Default workflow 119
6. What about the different crypto tokens? 119
7. Performance 119
7.1. PIII 850MHz, 256 MB RAM 119
17. Packaging 121
1. Common Notices 121
1.1. Required Perl modules 121
2. RPM-based system 121
2.1. RedHat/Feodora 121
2.2. SuSE 121
3. Debian 123
4. BSD 123
18. Software Design (legacy from design guide) 124
1. Database(s) 124
2. Interface construction 124
3. openca.cgi 124
4. libraries 124
5. modules 124
6. commands 124
7. Dataexchange and Node management 124
A. History 125
1. PKI Scenario before OpenCA 125
2. PKI and eGovernment 126
3. Internet Standards 126
4. The Project's Purposes 126
5. The Project's Achievements 127
6. The OpenCA Project 127
6.1. The project start 127
6.2. Offering Help to Other Projects: OpenSSL 128
6.3. CVS and Mailing Lists 128
6.4. The Open Source Choice 128
6.5. Migrating to SourceForge 129
B. Authors and Contributors 130

1. Michael Bell 130
2. Chris Covell 130
3. Massimiliano Pala 130
4. Ulrich Bathels 130
5. FAQ 130
C. FAQ 131
1. General PKI Issues 131
1.1. What is a certificate? 131
1.2. Which informations does a certificate contain? 131
1.3. What is a request? 131
1.4. Which information does a CSR contain? 131
1.5. What is a CA? 132
1.6. Why should I not place the CA on the same machine like the RA? 132
1.7. What is an extensions? 132
1.8. I use Windows 2000 and Internet Explorer 6 SP1 and it don't show any CSPs. 132
1.9. How can I setup a sub CA? 132
2. General OpenCA Issues 132
2.1. Does it be possible to revoke a certificate without any user interaction? 132
2.2. I try to add a role and get the message The role XYZ exists already! 133
2.3. All cryptographic operations fail. 133
2.4. Apache's error_log reports a nonexistent option -subj of openssl req 133
2.5. Apache's error_log contains a message from IBM DB2 that the environment is not setted 133
2.6. What do the new features of 0.9.2 be? 133
2.7. I try to approve and sign a request with Mozilla and it fails. 134
2.8. I try to approve and sign a request with Konqueror (KDE) and it fails. 134
2.9. How is the format of the disc to import the CA certificate from the root CA? 134
2.10. OpenSSL reports entry 1: invalid expiry date 134
2.11. Outlook cannot encrypt mail with imported certificate 134
2.12. My Outlook freezes after I received a signed email 134
2.13. General Error 6751 during certificate issuing 135
3. Installation Issues 135
3.1. FreeBSD and OpenCA 135
3.2. What is a hierarchy level? 136
3.3. Undefined subroutine &main::xyz 136
3.4. Symbolic link installaton failed 137
3.5. After the installation all common parts are missing 137
3.6. Conflicting Modules 137
3.7. The xml path to the access control is missing 138
3.8. Unknown Login Type 138
3.9. Type Mismatch during request generation with Internet Explorer 138
4. Configuration Issues 139
4.1. How can I configure my httpd.conf for virtual hosts? 139
4.2. How can I configure virtual hosts with ./configure? 139
4.3. I have some users which should not be published in LDAP. Does it be possible with OpenCA? 140
4.4. Does it be possible to authenticate users by their certificates at the apache before they will be authenticated by OpenCA itself? 140
4.5. I want to update my 0.9.2 installation. Is this dangerous? 141
4.6. I want update to 0.9.2. How can I update my sql database? 142
4.7. If I run openca-ocspd then I obtain a segmentation fault. 143
4.8. I installed a second public interface, run configure_etc.sh and now are all the paths in the other public interface wrong. 143
4.9. I issue a certificate for a mailserver but sendmail doesn't work and reports an errormessage which includes reason=unsupported certificate purpose 143

4.10. My (Microsoft) client hangs after it tries to start a secured connection 143
4.11. Outlook freezes when receiving a signed Mail but worked already fine for some days 144
4.12. During the request generation OpenCA fails and reports a too short textfield 144
4.13. Can I place my organization's logo on the web interface? 144
4.14. Microsoft Smartcard Logon 144
4.15. Cannot create new OpenCA tokenobject 147
4.16. How can I use a Luna token with OpenCA 0.9.1 147
4.17. How can I include a complete certificate chain into a PKCS#12 file? 148
4.18. Unknown login type 149
5. Access Control problems 149
5.1. Always get a login screen - again and again 149
5.2. Error 6251023: Aborting connection - you are using a wrong channel 149
5.3. Error 6251026: Aborting connection - you are using a wrong security protocol 150
5.4. Error 6251029: Aborting connection - you are using the wrong computer 150
5.5. Error 6251033: Aborting connection - you are using a wrong asymmetric cipher 150
5.6. Error 6251036: Aborting connection - you are using a too short asymmetric keylength 150
5.7. Error 6251039: Aborting connection - you are using a wrong symmetric cipher 150
5.8. Error 6251043: Aborting connection - you are using a too short symmetric keylength 150
6. Dataexchange 151
6.1. I try to export something but I get error 512 permission denied for /dev/fd0 151
6.2. I try to import the CA certificate but it doesn't work. 151
6.3. I crashed the database of the online server and now I want to import all data again. How can I do it? 152
6.4. I try to export the requests to the CA but it doesn't work 152
7. LDAP 153
7.1. Errormessage: Connection refused. 153
7.2. Errormessage: Bind failed. Errorcode 49. 153
7.3. The resultcode of the nodeinsertion was 65. 153
7.4. How can I get more debugging messages from OpenCA's LDAP code? 153
7.5. How can I get more debugging messages from OpenLDAP? 154
8. Internationalization 154
8.1. How can I fix a misspelling for a language? 154
8.2. How can I add a new langiage? 154
8.3. The compilation/make fails on the Perl module gettext 154
Bibliography 156
Glossary 158

List of Figures

1.1. Database oriented view 4
1.2. Logical data view 5
1.3. Complete technical overview 5
1.4. Lifecycle of objects 8
4.1. Passes of the accesscontrol 27
4.2. Passphrase based login 28
4.3. Tokenconcept 32
6.1. Phases of the CA initialization 71
6.2. Phase I of the CA initialization 72
6.3. Phase II of the CA initialization 74
6.4. Phase III of the CA initialization 75
7.1. Request a certificate 85
10.1. Example cryptolayer with tokens 95
11.1. Passes of the accesscontrol 97
11.2. Channel verification 98
11.3. Identification of the user 99
11.4. Access control list 100
15.1. LDAP source schema 112

List of Tables

15.1. Schema usage 111
15.2. Schema usage for user certificates 111
16.1. Default OpenCA workflow 119
16.2. 1000 User test 120
C.1. Texttypes for different databases 142

List of Examples

4.1. channel configuration 27
4.2. Login and Passphrase configuration 29
4.3. Authentication with certificates 29
4.4. Session configuration 30
4.5. Basic ACL configuration 30
4.6. Permission for serverInfo 31
4.7. Allow all 31
4.8. OpenSSL configuration - Authority Key Identifier 35
4.9. Minimal SSL client extensions 37
4.10. Minimal SSL server extensions 37
4.11. Minimal SMTP extensions for a single certificate 38
4.12. Additional attributes configuration 38
4.13. PKCS#10 configuration 39
4.14. Basic CSR configuration 40
4.15. Configuration example for a XML file based HTML-select 41
4.16. Download configuration 46
4.17. Export configuration 46
4.18. Local export configuration 46
4.19. /etc/mail/sendmail.mc 53
6.1. SSCEP configuration 82
15.1. suffix in ldap.xml 108
15.2. excluded roles in ldap.xml 109
16.1. batch_new_user.txt 116
16.2. batch_new_process.txt 117
16.3. batch_process_data.txt 117
17.1. SuSE packaging 122
C.1. General error 6751 during certificate issueing 135
C.2. Bad passphrase error log during certificate issueing 135
C.3. Full errormessage for missing functions 137
C.4. Already present symbolic link 137
C.5. Search for XML::Twig modules 138
C.6. Type Mismatch on Internet Explorer 139
C.7. virtual host configuration 139
C.8. ./configure and virtual hosts 140
C.9. Client authentication with mod_ssl 141
C.10. OCSP configuration for LDAP 143
C.11. OCSP configuration for http 143
C.12. Failed request upload 153

Introduction

Public Key Infrastructures (PKIs) are one of the most widely accepted musts of the future. The problem is that most applications can be secured with certificates and keys but it is really difficult and expensive to setup PKIs, the reason being that flexible trustcenter software (especially for Unix) is expensive. This was the starting point of OpenCA. Our goal is production of an open source trustcenter system to support the community with a good, inexpensive and future-proof solution for their base infrastructure.

OpenCA started in 1999. The first idea consisted of three major parts - a Perl web interface, an OpenSSL backend for the cryptographic operation and a database. This simple concept is still our motto today. Nearly all operations can be performed via some web interface. We have six pre-configured interfaces and many more can be created from them, depending on the need. The cryptographic backend is still OpenSSL, which is in no way a disadvantage. We want to build the organizational infrastructure for a PKI. This is our major goal and the guys from OpenSSL have much more experience with cryptography than we have. Our databases store all the needed information about the users' cryptographic objects like Certificate Signing Requests (CSRs), Certificates, Certificate Revocation Requests (CRRs) and Certificate Revocation Lists (CRLs).

If you imagine that the development of OpenCA will be finished in some weeks or months then you are probably on the wrong way. There are many items which still need to be implemented. Today we support the following elements (this is an incomplete list just to give you an impression of how complex the subject matter is):

- Public interface
- LDAP interface
- RA interface
- CA interface
- SCEP
- OCSP
- IP-filters for interfaces
- Passphrase based login
- Certificate based login (including smartcards)
- Role Based Access Control
- Flexible Certificate Subjects
- Flexible Certificate Extensions
- PIN based revocation
- Digital signature based revocation
- CRL issuing
- Warnings for soon to expire certificates
- support for nearly every (graphical) browser

OpenCA is designed for a distributed infrastructure. It can, not only handle an offline CA and an online RA, but using it you can build a whole hierarchy with three or more levels. OpenCA is not just a small solution for small and medium research facilities. The goal is to support maximum flexibility for big organizations like universities, grids and global companies.

The OpenCA guides consist of four parts: The first part is a design guide which should help you to setup a good infrastructure. The second part describes all the activities which must be performed offline by some administrator. The third part is the user guide which describes all the available features. The last part is the technology guide which documents the ideas behind OpenCA. This part is meant only for developers and hardcore administrators to understand what's going on.

Finally we wish to thank everybody who helped us program, test and document OpenCA. This also includes of all the universities and companies which finance the work of our developers.

Part I. Design Guide

Table of Contents

Preface iii
1. General Design 4
1. Basic Hierarchy 4
2. Interfaces 5
2.1. Node 6
2.2. CA 6
2.3. RA 6
2.4. LDAP 6
2.5. Pub 6
3. Configuration 7
4. Database 7
5. Interface 7
6. Lifecycle of the objects 7
7. Sub-Ca 8
7.1. Example 1 8
7.2. Example 2 8
2. Recommendations 9
1. Hardware Issues 9
1.1. Time 9
1.2. Failing disks 9
1.3. Hardwaremonitoring 9
2. Physical Security 10
2.1. Safes and Dataorganization 10
2.2. Buildings 10
3. Network Issues 10
4. Certificate Issues 11
4.1. CDPs 11
4.2. Application specific problems 12
5. Organizational Aspects 12
5.1. Dual Access Control 12
5.2. Privacy vs. Security 12
5.3. Enforcement of Access Control 12
5.4. Privacy Officer Integration 12
5.5. Enterprise Integration 13
5.6. Parallel use of several end user PKIs 13

Preface

The meaning of this guide has changed. It must be reorganized. It should now help people to implement a PKI which use the full power and flexibility of OpenCA. Mainly it should give the people an overview about the hierarchies which can be implemented with OpenCA.

Before you start installing or editing OpenCA you should understand the most important principles of OpenCA's design. The first section describes the general design and the second section describes the software design.

This guide will be incomplete at every time because it is the part of documentation which must changed at every time the software changes. So please be lenient toward us :)

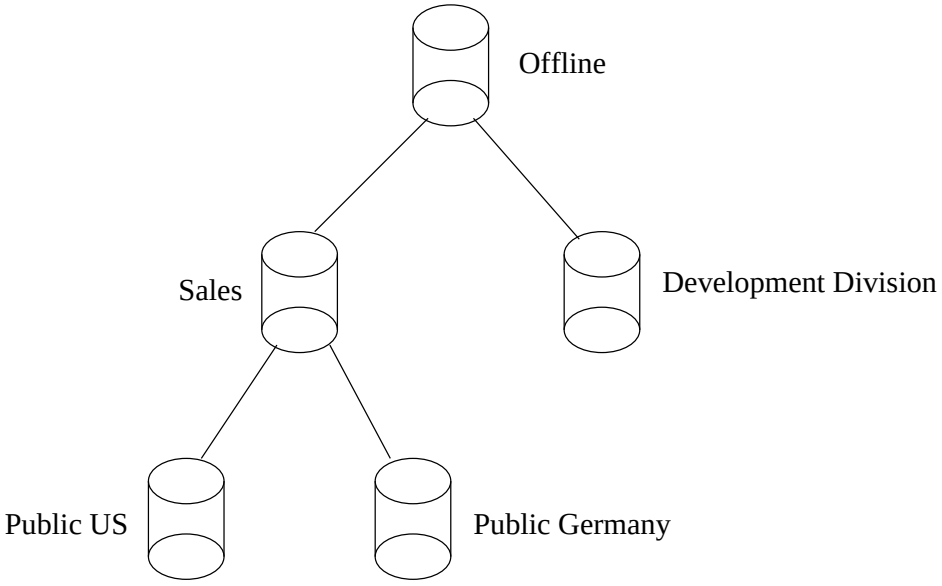
Chapter 1. General Design

We start here from scratch to give everybody a chance to understand how OpenCA works. So if you think about these boring guys who write this please take in mind that also OpenCA novices must have a chance to understand the software.

1. Basic Hierarchy

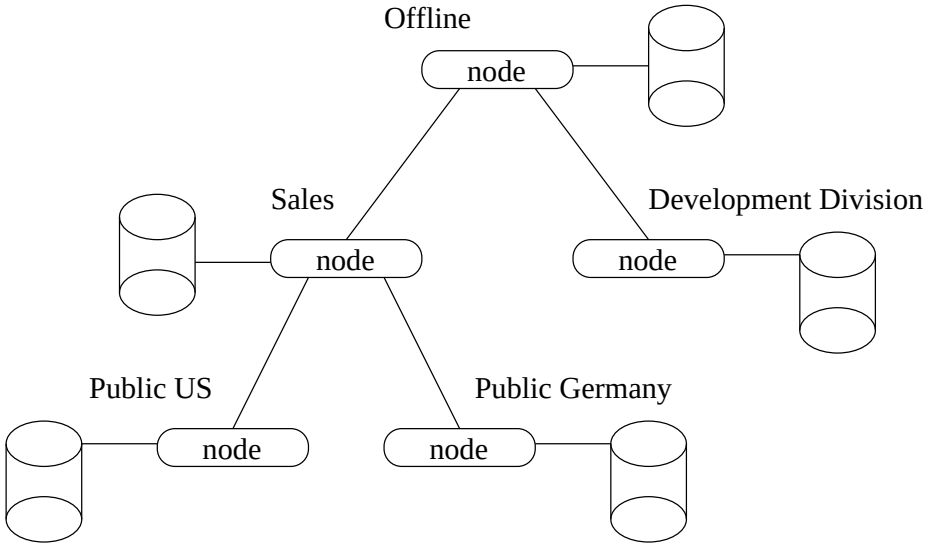
The basic idea of every X.509 PKI (Public Key Infrastructure) is a strong hierarchical organization. This results in a tree of databases if we try to create a distributed PKI architecture.

Figure 1.1. Database oriented view



The dataexchange between such isolated databases can be handled automatically if you use a distributed databasesystem but in the sense of OpenCA such a distributed databasesystem is only on database in our tree. If you have really isolated database (e.g. for an Offline CA) then you must have a technology for the dataexchange and the management of the complete node in the hierarchy. This management functionalities are bundled in an interface called node or node management. So the design of OpenCA looks like follows

Figure 1.2. Logical data view



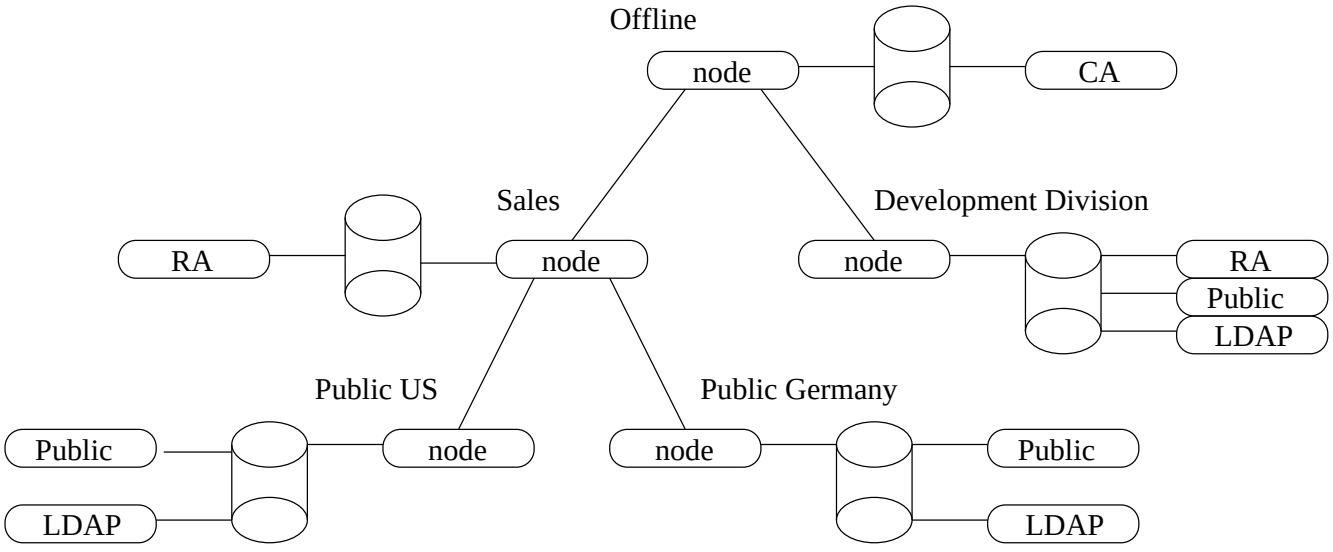
Normally every server in the infrastructure of the trustcenter has it's own database for security reasons. This hierarchy is the backbone of the trustcenter. \

2. Interfaces

After you know the basic infrastructure of OpenCA you possibly want to know what we think about such things like CA, RA, LDAP and a public interface which is sometimes called webgateway. OpenCA supports all these software-components via special web interfaces.

If you want to design a powerfull trustcenter then you must have a concept how you want to organize your workflow. You can see an example in the following picture.

Figure 1.3. Complete technical overview



OpenCA actually supports the following interfaces:

1. Node (for node management)
2. CA
3. RA
4. LDAP
5. Pub
6. SCEP

2.1. Node

This interface manages the database and handles all the export and import functionalities.

The database can be initialized what means that OpenCA can create all the tables but OpenCA cannot create the database itself because this differs for every vendor. So we need a database with the appropriate accessright and a new database. The interface includes some functions for the backup and recovery of such a node but please bear in mind that you MUST have a seperate backup of the CA's private key and certificate. There is no default mechanism in OpenCA to backup the private key. We don't implement it because first we found no general secure way to backup a private key and second the most CA's use HSMs and therefore you need a completely different and usually proprietary backup strategy.

The export and import will be handled by this interface too. You can configure different rules for the synchronization with nodes on a higher and a lower level of the hierarchy. This includes the configuration of the objects and status which can be exchanged. The configured filters avoid status injections from lower levels of the hierarchy.

2.2. CA

The CA interface has all function which you need create certificates and CRLs (Certificate Revocation Lists). The CA includes also all functions which you can use to change the configuration via a webinterface. It is not possible to change the configuration via another webinterface.

The CA is the home of the batchprocessors too. OpenCA includes some powerful batchprocessors to create certificates fullautomatically from ERP-systems (Enterprise Ressource Planning - e.g. SAP, HIS, NIS or /etc/passwd).

2.3. RA

OpenCA's RA is able to handle all kinds of requests. This include things like editing requests, approving requests, creating private keys with smartcards, delete wrong requests and email users.

2.4. LDAP

The LDAP interface was implemented to seperate the LDAP management completely from the rest of the software. This is necessary because there are many functions which are really special for LDAP admins too because only a few users need these features.

2.5. Pub

The Public interface include all the small things which the users need. This is only a small list and perhaps it is incomplete

- generates CSRs (certificate signing request) for Microsoft Internet Explorer

- generates CSRs for Mozilla 1.1+ and Netscape Communicator and Navigator
- generates clientindependent requests and private keys (e.g. for KDE's konqueror or server administrators which don't know how to create a private key and request)
- receives PEM-formatted PKCS\#10 requests from servers
- enrolls certificates
- enrolls CRLs
- supports two different methods revocation
- search certificates
- tests usercertificates in browsers (Microsoft Internet Explorer and Netscape Communicator and Navigator 4.7x)

3. Configuration

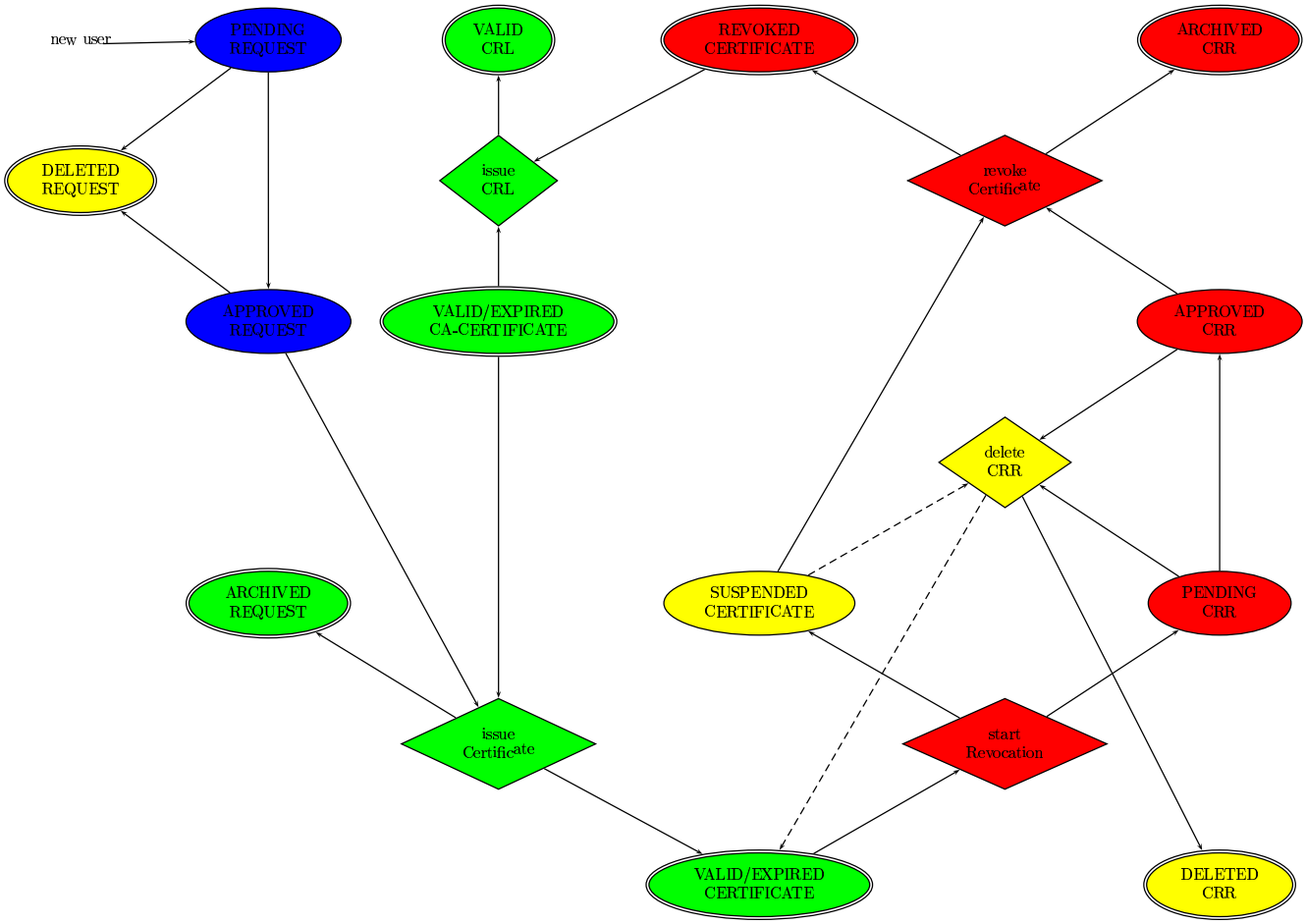
4. Database

5. Interface

6. Lifecycle of the objects

Sometimes there are confusions about the status of OpenCA's objects. Therefore you can check here the complete life-cycle of all OpenCA objects. If this figure is not complete, you find a mistake or you don't understand something then please contact openca-users@lists.sf.net.

Figure 1.4. Lifecycle of objects



7. Sub-Ca

7.1. Example 1

Direct chaining of the Root CA and Sub CA including an image.

7.2. Example 2

Chaining of a Root CA and a Sub CA via a normal public interface.

Chapter 2. Recommendations

This chapter should give you a lot of hints which you should bear in mind if you design your first PKI. Please don't ignore this section if you are an experienced PKI administrator. We try to list the big traps here. So if you know another major problem then please add it.

1. Hardware Issues

This section lists hardware issues which were a problem for some PKIs during it's productional use. This list does not discuss performance issues.

1.1. Time

One of the biggest problems of PKI systems is the time. There are two different kinds of computers - online and offline. The usual administrators logic is that a network connected computer can use a timeserver. The question is can you trust this timeserver? A timeserver creates two problems. First is the timestamp really from the timeserver and second is the time source of the timeserver trustworthy? The connection to a timeserver can be secured via tunnel technologies like IPsec but the real problem is the timesource. The most timeservers use finally a radio clock which receives a not signed timesignal from radiostation. This signal can be easily faked because it is very weak. So network timesources really insecure.

After the online computers lost all of their advantages we can discuss offline technologies. Radioclocks are problematical and so we have not to discuss them twice. Also many buildings made from good ferroconcrete has no problems with radiosignals because they implement really nice Faradeay cages. What we have for alternatives? First we need a trustworthy timesource. Simply take a digital watch and compare it's time with several other clocks in the internet, the videotext of your TV, a radioclock, the sun ;-), GPS and any other source you can find. Second you transfer the time from your watch to the computer. Last but not least you have to check the drift and the clock itself on the computer. The drift is a small and easy to handle problem. The clock itself is a much bigger problem. If your computer is always connected to a power outlet then the clock should only drift. Please remember this if you put your CA on a notebook and the notebook into a safe. Several new notebooks have really bad CMOS batteries which result in a wrong time at every reboot. You see time is not trivial.

1.2. Failing disks

The most common hardware crashs are cooler and disk failures. You should have of course a backup with all important data - especially with ALL issued certificates. Never lose a certificate or you must revoke the complete CA. Backups are a nice thing but it costs some time to recover from a backup. This results in two facts. First you must have a detailed (time-) plan how to recover from a backup. Second you should be able to tolerate diskcrashes. RAIDs are sometimes expensive but they helps a lot (ask you SAN admins :)).

1.3. Hardwaremonitoring

Usually you will see if your laptop crashes. A crashing offline computer can be detected by visual monitoring too. A crashing online component of a PKI is problem because important informations are not available. Such informations are new certificates and CRLs. Your services are offline too. This includes SCEP. If a public interface of the security infrastructure is down then you will get problems with your users in the future. So the trust into a running infrastructure is very fast gone. Please note that a simple ping is not enough. You cannot detect a crashed web or OpenCA perl server with a ping. Software bugs can also cause 100 percent load. I know this problem from our web mail programs really well.

2. Physical Security

2.1. Safes and Dataorganization

If your offline CA consists of one offline notebook and you want to ensure dual control and no single point of failure then you can do it for example with one IT module safe and two data safes. An IT module safe is a safe with own climatisation and UPC which has the same physical protection level like a safe. This safe is used to allow nonstop operation of the notebook which reduces time and availability problems. All three safes should have two locks. This ensures dual access control by key sharing. This is really simple and really efficient too.

The organization of the safes is really easy. You split the CA passphrase into two parts - front and back. The organization of the data and computers is like follows:

1. The IT module safe includes the notebook (with the CA private key) and the front part of the passphrase.
2. The first data safe includes the backups (including the private key backups) and the back part of the passphrase.
3. The second data safe includes the front and back parts of the passphrase.

This organization ensures that one broken safe doesn't corrupt the infrastructure. It is important to start immediately a rollout of a new infrastructure but there is no reason for panic. This arrangement also ensures that a lost of one safe doesn't stop your operation. You need two safes to survive and a lost of one safe is acceptable at minimum for a short period of time.

Please remember that this is a really simple idea for medium risk CAs. High risk CAs should be use more complex schemes to not only tolerate one broken safe. They should be able to tolerate at minimum two broken safes to have a longer schedule for the roll-out of a new CA.

2.2. Buildings

This is more an area for a facility manager. The rooms with the PKI safes inside should have for solid walls and a door with two locks. The room including the climatisation system should be fire safe for 90 minutes (F90). The room and the entry should be camera observed. The cameras in the room itself should show the persons but not the keyboards and monitors. Papers should not be readable. The recorder for the camera should record one week at minimum. The room needs an alarm system. The room must be safe against electro magnetic pulse (EMP) and water too. This is only a short notice. Please ask some assurance specialists or architects for more details.

3. Network Issues

A PKI is only fully operational if all services of the PKI are fully operational. This include things like OCSP and SCEP but it includes the public gateway too. Many people think it is enough if OCSP and one CDP still works but this is wring. The first thing is that the most applications don't understand OCSP. The second problem is that the last running CDP perhaps only support LDAP but there are applications which only supports HTTP and even more problematic is single running HTTPS CDP. The core mistake in this assumption is the meaning of fully operational. A PKI is not fully operational if only the CDPs still work. If nobody can download a new certificate or a certificate of a user which he never contacts before the PKI service is interupted. The PKI is still secure but many people mix up secure and operational.

In a time of server consolidation and omnipresent networks it is important to understand that all public PKI services must be available after a single failure. This includes network and power outages. A second fibre only helps if it is not in same pipe like the other one. a digger don't differs between the fibres if it cuts a pipe - I know this situation really

well :(If you have big distributed units then it is recommended that at minimum two of these units run the public interfaces. In this case you should have independent interconnects.

4. Certificate Issues

Today this section consists of two big groups of problems - CDPs and application specific problems. First we discuss all the CDP stuff and second we discuss the different applications. Several applications are discussed in the other OpenCA guides too if we developed special configurations to solve the problem.

4.1. CDPs

CRL Distribution Points are a critical area. THere were several PKIs in the first years which have no CDPs or only one CDP. Secure applications must verify the state of a used certificate. Such applications check the CDP field in the extensions of a certificate to find a usable verification source. Today a CRL Distribution Point has not to be a source for a CRL. It can be an OCSP responder too. A CDP is today more a Certificate Status Point.

The first question is which protocols must be supported. The common procols LDAP and HTTP should be supported always. HTTP is supported by nearly all devices but some devices like LDAP more than HTTP - especially some network solutions. Additionally there are OCSP and HTTP. OCSP is a protocol to verify the state of a single certificate. This protocol has a much better performance for slow network interconnects if you have large installations with many (revoked) certificates. HTTPS supports you with a trusted source but is a trusted source necessary for CRLs? Usually no but sometimes yes. CRLs are signed by the CA and they have period of validity. There is only one working attack. If a certificate was revoked, the old CRL is still valid and the CDP uses HTTP then an attacker could present the user the old CRL. So this attack is more a question of security policy than a technical question. The validity period is a policy question too because the most application don't accept CRLs if the timestamp for the next update is over. The cool aspect is the fact that a CRL is never invalid. There can only be a newer CRL but like mentioned this is more a political discussion.

The second question is how many physical CDPs do you need. If you design a scalable infrastructure then you must be able to tolerate service interruptions of several compontents like cables, routers, switches and servers. It is a really good idea to have always a local CDP if you have only a single interconnect to your central network but otherwise not every branch office need it's own CDP because if there computers including their file and mail systems are offline then it is not necessary to have a CDP online.

The third question is a really trickey one. Which order should the CDPs have? The question is interesting in the case of network outages. If you support LDAP and HTTP, you have three servers which all serve both protocols, the CDP are ordered like the servers and server one and two are not available because of a more or less big digger then your applications waits for four timouts until it can access the first CDP successfully. The application has to wait for timeouts from LDAP on server one, HTTP on server one, LDAP on server two and HTTP on server three. So it is a really good idea to mix the order of protocols and servers. This is most important for application which understand many protocols. Stupid alphabetical ordering using the DNS names is not recommended or do you think a user waits two minutes for the verification of every recipient of an email if he wants to send a commit for a meeting?

The last question is today often a not hundert percent solvable problem. Can a supported and necessary protocol crash other applications? If you use HTTPS then you should invest some time on this problem because it is a typical hen and egg problem. If you have Microsoft client and the client tries to verify the certificate status via a HTTPS CDP then it receives a certificate from the HTTPS server. This certificate must be verified. If the certificate is from the same CA then the client contact the HTTPS CDP again to verify the certificate and the verification problem starts again. This is a nice method to implement a busy wait. The only solution is to get a server certificate from another authority which use no CDPs or at minimum no HTTPS CDPs. The insight is that you should be really careful with the usage of crypto protected CDPs.

4.2. Application specific problems

4.2.1. Mailservers

If you start thinking about emailsecurity then the starting point is S/MIME or PGP but sometimes the admin know ethereal and they continue with IMAPS and POPS. So far so good. Later they think about server based security and try to implement SMTP over TLS. OpenCA issues certificates which can be used for SMTP over TLS because the certificates act as client and server certificates. If you use other software than OpenCA or you create a new role for such servers then you should read the documentation of your SMPT server really carefully.

A SMTP server can use two methods for SMTP over TLS. It can use one certificate which includes the extensions for TLS client and servers or it can use two certificates - one client and one server certificate. Please read the specs carefully and then read specifications for TLS clients and TLS servers. If the SMTP server reports an error after **start tls** then there is usually a missing extension or a problem with the certificate chain. See Section 3.2.2, “SMTP server”.

4.2.2. Netscape clients

Old netscape clients are not conform with the RFC standards. They use the common name as a regex to match the servername. They ignore the subject alternative name completely. A workaround is described in the configuration area of OpenSSL (see Section 3.2.1, “HTTPS server”).

4.2.3. OpenLDAP

OpenLDAP is an open source software but it's TLS implementation is not the best one. It doesn't check the subject alternative name first. First it checks the common name of the subject to match the DNS name or IP address. This fails of course if the certificate subject inlcudes a regex for old Netscape clients. Two non standard compliant softwares collidate here. Netscape needs regular expressions, OpenLDAP don't support regular expressions and both softwares doesn't check the subject alternative name first. So it's your job to made a tradeoff or to use Mozilla.

5. Organizational Aspects

5.1. Dual Access Control

dual access control (physical and technical but first organizational apsects)

5.2. Privacy vs. Security

privacy vs. security (e.g. camera recorders)

5.3. Enforcement of Access Control

access control but who controls the access control regulations (no self control)

5.4. Privacy Officer Integration

how to integrate privacy officers public certificates - not always public certificates - which fields must be published? PID vs. new ID what is the identity of a person in convetional areas

5.5. Enterprise Integration

already existing ERP DBs metadirectories

5.6. Parallel use of several end user PKIs

There are numerous situations where it is a good idea to operate more than one PKI for endusers. Perhaps you need a server CA and a seperate user CA. Sometimes an old CA is still active by issueing CRLs because there are still valid certificates but the new CA already issues certificates. Other people using different CAs to establish an easy access control by the certificate chains (so called trust paths). You see there really many situations where you have to operate more than one PKI.

The most PKI programmers like me have no problem to distinguish between different PKIs because we always ask who issues this certificate but which normal user do this. He simply looks at the certificate, calls the hotline and asks why the certificate for Jon Doe with the serial 12345 does not work. The guy from the hotline looks into it's computer and answers that the certificate is correct and valid. So what's going on?

A certificate has two significant things to identify a certificate which are different from the common name in the subject of the certificate and which are easy to handle (by the way keyID and issuer from the authority key identififer are not easy to handle for an enduser). First there is a serial and second there is an issuer. If a customer calls a hotline then the easiest way to handle a problem are organization wide unique serials. If you start a second CA or you have to repalce an old CA never reuse serials if this is possible. You will search for hours if somebody calls you and reports a broken certificate chain for certificate 12345 and you two of those certificates. If you ever issued certificates with identical serials then always asks for the issuer if you receive an error report. Never ever create a replacement for an old CA with the same name. It only course trouble.

The resume is very simple. If you avoid duplicate identifiers then automatically avoid many problems.

Part II. Installation and Configuration Guide

Table of Contents

Preface xvii
3. Installation 18
1. Preparations 18
1.1. Software 18
1.2. Hardware 18
2. Configure 18
2.1. Host System Configuration 19
2.2. Filesystem paths 19
2.3. Webserver specific stuff 20
2.4. Email 20
2.5. Compiling features 20
3. Installation 21
4. config.xml (for RPMs and DEBs too) 21
4.1. Configuration sections of config.xml 22
4.2. How to setup two management interfaces on one server? 24
4. Configuration 26
1. Access Control 26
1.1. Channel verification 27
1.2. Login 27
1.3. Session management 29
1.4. ACLs 30
2. Token and keyconfiguration 32
2.1. OpenSSL 33
2.2. Empty 34
2.3. LunaCA3 34
3. OpenSSL 34
3.1. Certificate Extensions 34
3.2. Profiles 36
4. CSRs 38
4.1. Additional Attributes 38
4.2. PKCS#10 Requests 39
4.3. Basic CSR 39
4.4. SCEP 41
5. Subject 41
5.1. Common stuff 41
5.2. dc style 42
6. Subject Alternative Name 43
7. LDAP 44
7.1. Configuration of the Directory 44
7.2. Configuration of the online components 44
7.3. Writing Certifciates to the Directory 44
8. SCEP 45
8.1. OPENCADIR/etc/servers/scep.conf 45
8.2. OPENCADIR/etc/config.xml 45
9. Dataexchange 45
9.1. Configuration 45
9.2. Adding a new node 49
10. Databases 49
10.1. PostgreSQL 49
10.2. DBM Files 51
11. Email 52

11.1. Sendmail with basic SMTP authentication 52

Preface

This guide should describe all installation and administration issues of OpenCA. Some answers are perhaps in the FAQ but every detail which you can find in our docs about the istallation you can find here.

Chapter 3. Installation

1. Preparations

1.1. Software

OpenCA is not a complete monolithic system. It uses several software products from other developers of the Open Source community. The following things are used:

- Apache
- mod_ssl
- OpenSSL
- OpenLDAP
- Perl

We use a lot of different Perl modules. Usually OpenCA installs all these needed modules. Sometimes we have problems with already installed modules.

- Digest::MD5 is usually problematical on modern Linux distros because Perl 5.8 includes Digest::MD5 already and some old version of Digest::MD5 has big problems with new Perl. You can simple outcomment Digest::MD5 in src/modules/Makefile.
- Gettext causes sometimes trouble because version 1.0.0 has a different export behaviour than 1.0.1. The result of this small "bug" is that OpenCA doesn't work with gettext 1.0.0. Please update your gettext or remove it from the system.
- CGI::Session is sometimes problematical if you use OpenCA's access control. If you login then OpenCA sends you a cookie and write some session data to the disk. Old version of CGI::Session doesn't flush the data automatically and so the data files on the server disk are empty. OpenCA::AC includes some fixes but it is strongly recommended to have only CGI::Session 3.92+ on your system.

1.2. Hardware

OpenCA was tested on several softwarearchitectures but not on so many hardware architectures. Therefore we publish a list of used hardware. Please remember that OpenCA can be used on any system which support Apache, mod_ssl, OpenSSL and Perl. So if you have Unix box then it is usually possible to run an OpenCA on it.

- i386 with Linux, FreeBSD, OpenBSD and NetBSD
- UltraSparc with Solaris 8 and Linux
- PowerPC with AIX

2. Configure

OpenCA uses the usual Open Source method to **configure** the source. We only use **configure** to compile and install the software but we don't use **configure** for the configuration of the installed system. **configure** make some defaults settings but the real configuration is described in the post-install section.

We will describe the ideas and options in the next section grouped by such things like path settings, mail, web-server related stuff. If you don't understand an explanation then please contact <openca-user@lists.sf.org>. The install options are now lesser because we changed the installation process from 0.9.1 to 0.9.2 to get usable packages and better internationalization.

We don't document the general options of **configure** because it is not our job to document autoconf. We will only describe OpenCA specific options.

2.1. Host System Configuration

You should define the used system before you start configureing OpenCA itself. OpenCA must know several parameters about your system to work properly.

`--with-openssl-prefix=`*`DIR`*

Usually OpenSSL is present on the most Unix systems because it is the best available Open Source cryptotoolkit. The problem is that several old distributions only include support for OpenSSL 0.9.6 but OpenCA needs version 0.9.7.

If you install an OpenSSL from source then it installs in `/usr/local/ssl`. This is the directory which you must specify. If you system already includes a proper version then you have not to use this option or you can enter `/usr` on the most linux boxes.

`--with-language=`*`LANGUAGE`*

If you install OpenCA on a server with a special language setting or you want to use this server with a special language interface then you must specify the locale. OpenCA supports today `de_DE`, `es_ES`, `fr_FR` and `it_IT`.

`--with-openca-user=`*`ARG`*

OpenCA installs several files which should not be owned by the webserver user. Usually the owner can be root or special OpenCA user. It is recommended to use another user than root.

`--with-openca-group=`*`ARG`*

OpenCA installs several files which should not be owned by the webserver group. Usually the group can be root or special OpenCA group. It is recommended to use another user than root. If you install several CA you can setup a group openca or *`pki`* for *`example`*.

2.2. Filesystem paths

We have three different groups of paths - common stuff, prefixes for the different components of OpenCA and the paths for files of the webserver.

One path cannot be classified - `--with-module-prefix=`*`DIR`*. This path can be used to put all Perlmodules which OpenCA installs in one directory to be able to remove OpenCA from your system without any residues. It is also a good idea to use this option if you need different OpenCA installations with different versions of OpenCA on your system. Later versions of OpenCA can have different modules with different interfaces which are not backwards compatible.

2.2.1. Common Prefixes

OpenCA includes a directorystructure to store all relevant data in one central place. This place can be specified with `--with-openca-prefix`. This installation option is recommended for normal installations from the source code. Secure or not the most users want to install packages (e.g. RPM or DEB). Packages have the big advantage that you remove or add a software without any risks. In this case we have to support the package maintainers with configuration options to build packages which conform with the guidelines for the distros. Therefore you can use `--with-etc-prefix`, `--with-lib-prefix` and `--with-var-prefix` too.

2.2.2. Component Prefixes

Today there are six different components - ca, ra, ldap, pub, node and scep. Every component must have a different name to have distinguished configuration files and distinguished paths. All the names will be calculated automatically. You have only to edit these prefixes if you need a special configuration like a second RA on the same machine.

2.3. Webserver specific stuff

The webserver configuration is the most complex and most simple part of the configuration too. If you have single http-server for OpenCA then you only need four options to configure OpenCA for this server. If you have a full featured corporate portal then you can integrate this software seemlessly in the the server. Therefore you can configure a lot of details. So we hope you find a good tradeoff ;-)

2.3.1. Common server informations

Every webserver needs some basic informations. These informations are the hostname (`--with-web-host`), the user (`--with-httpd-user`) and the group of the server (`--with-httpd-group`). These are the rudimentary informations which OpenCA needs before you can start configuring the paths. The defaults are an empty hostname, `nobody` and `nogroup`.

The most trivial installation case is the default apache installation. In this case you have only to set `--with-httpd-fs-prefix` to the directory where your apache is. All other directories will be set automtically.

2.3.2. Filesystem Paths

The standard webserver doesn't use Apache's default installation. Therefore it is possible to configure every detail of the installation. The first splitting is into CGI (`--with-cgi-fs-prefix`) and HTDOCS (`--with-htdocs-fs-prefix`). The most test systems don't need the other options. They have only to know where the appropriate directories are.

Our software was designed for really big companies and organizations too. They have usually portals for their employees and customers. If you have to integrate an OpenCA interface into such a portal then there are good news for you - you don't have to edit paths and links by hand. You can configure the placement of CGI and HTDOCS area of every interface seperately. The options are `--with-(ca|ra|ldap|pub|node|scep)-(cgi|htdocs)-fs-prefix`). We think that more flexibility is not necessary. So if you think OpenCA is to unflexible then write a mail to us with your ideas.

2.3.3. URL Paths

OpenCA 0.9.1 supports a lot of options to configure the URLs like the filesystem paths. This is possible with OpenCA 0.9.2 too but it is deprecated to do this with **configure**. Please read the post-install section. It can happen that these options will be removed from configure.

2.4. Email

The mailoptions are deprecated too. Please read the post-install section to understand how to configure mail. Please don't use the configure option because they can be removed in the next releases.

2.5. Compiling features

You can enable three extra features for compilation and installation. SCEP and OCSP can be enabled because they are extra softwarepackages which can work independently from OpenCA but they are included in the distribution. The

option `--enable-package-build` is used to support package maintainers. If it is activated all common parts of OpenCA are not installed automatically. This allows packagers to build seperate conflict free packages for every interfaces because all Perl modules and the common stuff can be put into seperate packages.

3. Installation

First run **make** and then run the different install commands. **make ca** and **make ext** are the same like **make** You have the following install options for **make**.

- `install-ca`
- `install-common` is only interesting for package maintainers because they can install the common stuff seperately from the rest.
- `install-ext` is the same like install online. This install target is deprecated.
- `install-ldap`
- `install-node`
- `install-offline` installs ca and node
- `install-online` installs ra, ldap, pub, scep and node
- `install-pub`
- `install-ra`
- `install-scep`
- `install-docs`

OpenCA incudes a startup script for it's daemons. The script is named `OPENCADIR/etc/openca_start`. The script starts the XML cache and the main server loop of OpenCA. Please remember to run this script after every boot operation. It is recommended to integrate a script `openca` to the appropriate runlevel.

4. config.xml (for RPMs and DEBs too)

After the installation all necessary files are in the correct directories but there are hunderts of files called `*.template`. These files contain placeholders which can be configured in `OPENCADIR/etc/config.xml`. Before you start using OpenCA check this file and run **OPENCADIR/etc/configure_etc.sh**.

OPENCADIR/etc/configure_etc.sh loads `OPENCADIR/etc/config.xml` and creates the correct files. If you use packages from distribution then OPENCADIR is usually empty because they create a directory `/etc/openca`.

`config.xml` contains seven big sections which will be described first. Second we describe how to setup an installation with only one common area but two management interfaces. This is useful if you want to test the dataexchange.

4.1. Configuration sections of `config.xml`

4.1.1. General options

Here you have to define some options which are relevant for several interfaces. The ca locality, organization and country affects the distinguished name and the preconfiguration of the LDAP stuff. Nevertheless you should read the LDAP section too. The values which you enter are directly used for `l`, `o` and `c`.

The mailpart is used for the node and RA interface. The `sendmail` field defines the command which will be used to send mails. You must have a mailprogram with a sendmail interface (e.g. postfix). You can enter every program which works like **sendmail -n**. There are several people which like postfix more than sendmail and we don't like do decide which mailprogram is the best one. The option `send_mail_automatic` configures the node interface. If the value is *YES* then OpenCA sends all incoming mails during an import automatically. This can be nice but it is dangerous too if you make a mistake. The `service_mail_account` is used as `From` for all sended mails. Usually this should be something like `<pki@your_org.edu>`.

The last option `policy_link` defines a link to your policy. It is highly recommended to don't ignore this value. If you have such a reference then you can modify the page request_success.html (add a hint) and the users can read all about the PKI at every time they want. If you have no such link then you receive dozens questions which are really simple but cost a lot of time and you have no base for your operations. Ok, I think I have not to explain the advantages of a policy here ...

4.1.2. web server configuration

Sometimes you need to run a CA on really unusual ports or you have to use https. It is also a little bit difficult for us to guess your correct hostname. Therefore you can specify these parameters in `config.xml`. The `httpd_port` should be the default port of the protocol and in this case it can be empty. If you need to run for example a http server on port 8080 then you have to use the option. Please remember to set the colons if you specify a port.

CRL distribution points (CDPs) can be specified extra. This is necessary because there are softwares which has problems with https in general or other softwares which try to download a CRL and before they start the download they want to check the CRL of the webserver but the CRL is not present (or why should somebody tries to download it :)) and so an endless loop starts - Microsoft CAPI is such a software.

If you setup a real CA then it is highly recommended to edit all files in `OPENCADIR/etc/openssl/extfiles` and `OPENCADIR/etc/openssl.cnf` too. Every certificate should contain at minimum two CDPs. It is best practice to have two http CDPs and two ldap CDPs. Such a solution allows fast migration and a good reliability.

4.1.3. ldap server configuration

Before you start working with OpenCA's LDAP code please be sure that your LDAP server knows the objectclasses `pkiUser`, `pkiCA` and `uniquelyIdentifiedUser`. The last objectclass was introduced by Entrust Technolgies to have a clean way to include serialnumbers into the subject of the certificate. Yes, it is proprietary but there is no other way to do it.

The following list is identical with the listof the list in the tech guide where you can find more informations about OpenCA's LDAP code and how to configure the details in the configuration files.

useLDAP	If you set this option to "yes" then the LDAP code will be activated.
update_ldap_automatic	If you want that the LDAP server will be updated during the import from a higher level of the hierarchy then you must set this option to `YES`.
ldap_host	This is the hostname of your LDAP server.

ldap_port	This is the port where your LDAP server listens.
ldaproot	The bind DN of the user which OpenCA uses to add data to the server.
ldaprootpwd	The passphrase for OpenCA's ldap account.

4.1.4. database configuration

First you have to decide which database module you want to use. OpenCA supports two modules - one for DBM files and one for SQL databases. `DB` activates support for DBM files and `DBI` activates the SQL support.

If you want to use SQL databases then you have to setup some additional parameters:

db_type	This is the type of the DBD driver. We support Pg, MySQL, Oracle and DB2. If you need support for another database then please contact us.
db_name	name of the database which OpenCA should use
db_host	host of the database but sometimes the drivers don't need the host.
db_port	same as for host
db_user	the database user for OpenCA
db_passwd	has not to be explained :)

4.1.5. module configuration

OpenCA has a mechanism to isolate the different interfaces from eachother to avoid conflicts especially double serial-numbers. The `module_shift` is the number of bits reserved for the IDs. You can use IDs from 0 to (2^`module_shift` - 1). 0 is the ID of the CA. All the other `_module_id`s must be in the scope of the module shift. Please be careful you cannot change the `module_shift` after the first definition.

4.1.6. configuration of relative paths

The `_url_prefix` options define the exact positions in the webserver. This depends highly on the positions of the files in the filesystem but you can configure aliases in the `httpd.conf`. So OpenCA is fully flexible.

4.1.7. configuration of SCEP

SCEP is really simple to configure but please don't forget the access control configuration. It is strongly recommended to restrict the source addresses which can access the SCEP server.

SCEP_RA_KEY	This is the PEM encoded private key of the SCEP interface. It has the same format like for mod_ssl.
SCEP_RA_CERT	This is the PEM encoded certificate of the SCEP interface. It has the same format like for mod_ssl.
SCEP_RA_PASSWD	This is the passphrase for the private key of the SCEP server. If you use a not encrypted private key (what is not recommended - then please set an empty string here. interface. It has the same format like for mod_ssl.

4.1.8. HSM support

OpenCA 0.9.2+ only supports Chrysalis-ITS Luna CA3 by default. If you have another HSM or TRSM then please contact us. It is really easy for us to support other security hardware which is supported by OpenSSL but we don't like to announce untested security hardware in such areas like PKI.

openssl_engine	This is the name of the OpenSSL engine. If you want to add an option like `-keyform` then please use this style: *`lunaca`* `-keyform` *`e`*.
hsm_utility	This must be the absolute path to the LunaCA utility.
hsm_slot	This is the slot id of the Luna CA3.
appid	This is the application id of the Luna CA3.

4.1.9. Dataexchange

The configuration of the dataexchange is really complex in OpenCA. You can find a description in the section about the configuration of the dataexchange (see Section 9, “Dataexchange”). If it is your first OpenCA installation then please use one of the templates. If you setup a production level PKI then you must understand this configuration before you use it. This is one of the most important configuration options to guarantee the security of the PKI.

4.2. How to setup two management interfaces on one server?

Before the explanations start please notice that it is important to first install the online components and then the offline components if you follow the instructions because the configuration of the offline components take care about the already configured onlinei components.

4.2.1. Online Components

The first installation uses only the normal steps - **./configure --with-your-options**, **make**, **make install-online**, edit `OPENCADIR/etc/config.xml` and **OPENCADIR/etc/configure_etc.sh**. Please use your options to configure the software and use the hierarchy level *`ra`*.

4.2.2. Offline Components

The first step is the protection of the already installed configuration files. Please set no permissions to the later needed configuration files in `OPENCADIR/etc/servers`.

```
chmod 000 etc/servers/*.conf*
```

The first four steps are the same as for the online components except of the configuration options where you should change at minimum the hierarchy level to *`CA`*. So first you do **./configure --with-your-options**, **make**, **make install-offline** and edit `OPENCADIR/etc/config.xml`.

The next step is really important you have to edit the file `etc/configure_etc.sh`. The directory with the serverconfigurations is protected because of the first step but all the other directories should only contain configuration files of the ca. Usually there should be the following directories:

```
/Test/OpenCA/etc/
/Test/OpenCA/lib/servers/ca_node
/Test/OpenCA/lib/servers/ca
/Test/htdocs/ca
/Test/htdocs/ca_node
```

After you fixed the script please run it. Now the offline components are installed and configured.

4.2.3. OPENCADIR/etc/menu.xml

`menu.xml` must be fixed manually because it includes only a basic configuration. You have to copy a complete `menu` section. The section must be renamed from *`ca_node`* to *`ra_node`*. The `cgi_prefix` must be fixed too. Please verifies the menus with the `name`s *`ra`*, *`ldap`* and *`pub`* to use the correct links to the node interface. If all values are correct then you have now a working testinstallation with two management interfaces.

Chapter 4. Configuration

1. Access Control

Before we begin with the access control itself some introductorily notes about the module ID. Every module in OpenCA has a module-ID. This ID you can find in the configurationfile of a module. The ID is used to create unique serial numbers for the requests. The moduleshift defines how many bits at the beginning of a serial number (the least significant bits) are reserved for the module's ID. The advantage is that you can issue a request at any module of OpenCA without synchronizing the databases at every time you issue a request because the module's Id is part of every request serial. The parameter in the configurationfiles are `ModuleID` and `ModuleShift`. You can configure both parameters via ./configure. The options are `--with-module-shift`, `--with-ra-module-id` and `--with-pub-module-id`. The ID of the CA is at every time zero.

OpenCA includes since version 0.9.2 a very powerful access control. The configuration of this access control is completely XML based. The configuration files are placed in `OPENCADIR/etc/access_control`.The document root element is openca and the complete configuration is placed in the area of the access_control element.

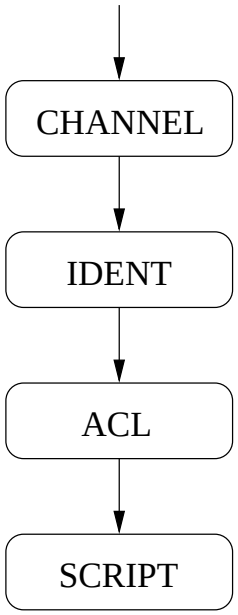
```
<openca>
    <access_control>
        The complete configuration should be here.
    </access_control>
</openca>
```

The complete system consists of four parts:

1. channel verification
2. login
3. session management
4. ACLs

Every step is a completely isolated pass except of the second and the third step which are unified in the second pass.

Figure 4.1. Passes of the accesscontrol



If you are looking for a more detailed description then please read the tech guide.

1.1. Channel verification

The channel configuration checks the parameters of the incoming connection to detect misconfigured apaches and obsolete clients. The values in the configuration are regular expressions except of the type. The type defines the type of the environmentvariables which should be tested. Today we support only mod_ssl.

If you use encrypted connection then you must use `ssl` as protocol. If you need an unencrypted connection like on the CA for the interfaces `ca` and `ca_node` then you must use `http` as protocol if you use mod_ssl. If you use an Apache without mod_ssl then you must use `.*` to match all incoming protocols. Please remember to set all the keylengths to 0 because otherwise the access control rejects all incoming connections.

Example 4.1. channel configuration

```
<channel>
    <type>mod_ssl</type>
    <protocol>ssl</protocol>
    <source>192.168.0.1</source>
    <asymmetric_cipher>.*</asymmetric_cipher>
    <asymmetric_keylength>0</asymmetric_keylength>
    <symmetric_cipher>.*</symmetric_cipher>
    <symmetric_keylength>128</symmetric_keylength>
</channel>
```

1.2. Login

We implemented three different ways to login to OpenCA:

- none

- passwd
- x509

1.2.1. none

The first possibility means that there is no login and everybody who pass the channel verification can use the interface. This possibility is nothing else than the deactivation of the access control.

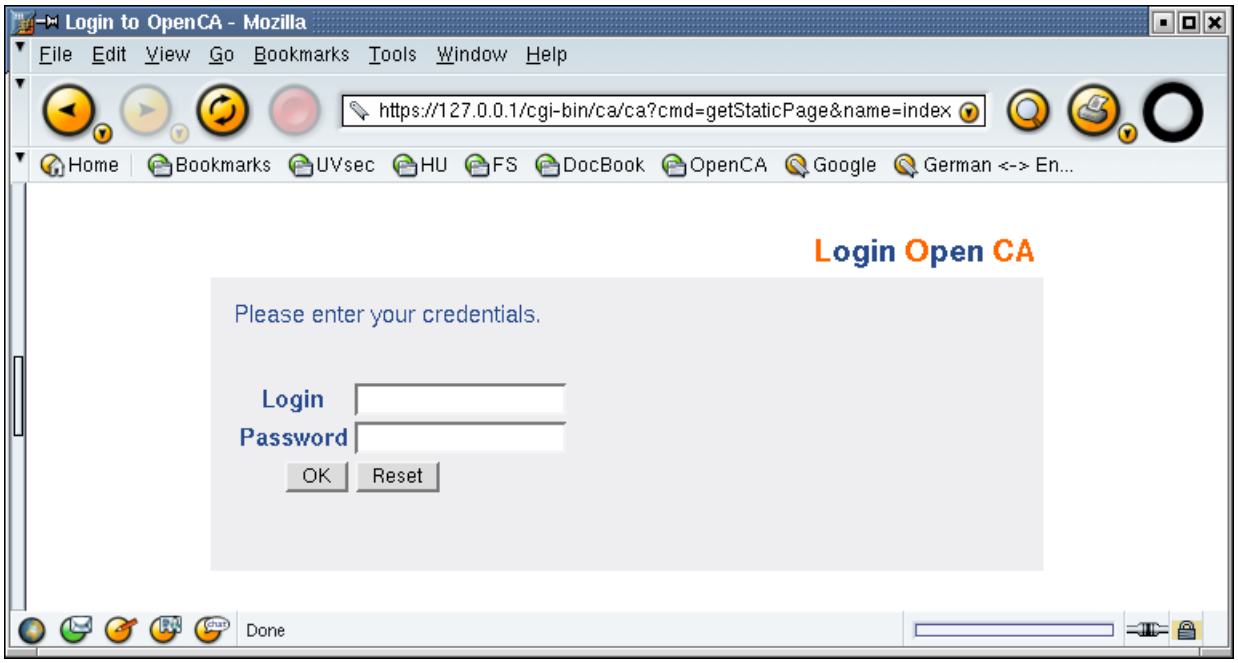
This is not only an option for debugging and testing. You can also use this option if you want to use a RA interface on a server which allows only RA access from the local machine but not over a remote computer.

```
<login>
    <type>none</type>
</login>
```

1.2.2. passwd

This method can be used to login via login and passphrase. OpenCA supports until now only the internal database in the access control configuration.

Figure 4.2. Passphrase based login



There you can specify several users. Every user has name, algorithm and digest. The algorithm specify which kind of digest should be used to hash the passphrase. OpenCA supports three algorithms SHA1, MD5 and crypt.

Example 4.2. Login and Passphrase configuration

```
<login>
    <type>passwd</type>
    <database>internal</database>
    <passwd>
        <user>
            <name>root</name>
            <algorithm>sha1</algorithm>
            <digest>3Hbp8MAAbo+RngxRXGbbujmC94U</digest>
        </user>
        <user>...</user>
        ...
    </passwd>
</login>
```

Before somebody tries to crack this hash the passphrase is root and this is the default passphrase of OpenCA :)

We prepared a script to generate the digests. The name of the script is openca-digest and it will be installed during make install*. The program is used like "openca-digest (help|sha1|crypt|md5) string".

1.2.3. x509

This is perhaps the most advanced method to login. The user must sign it's assigned session ID and the access control verifies the signature. The login name is the serial of the certificate because it is the only unique item in a certificate. The configuration is really simple. You have to set the position of the CA chain and that's all.

Example 4.3. Authentication with certificates

```
<login
    <type>x509</type>
    <chain>OPENCADIR/var/crypto/chain</chain>
</login>
```

The filtering of the users is not the job of the login because the login has only to identify the user. The filtering has to be implemented in the ACLs. Therefore we cannot recommend the x509 (smartcard) authentication until now.

1.3. Session management

The session management is today really simple to configure because we only support one method to manage a session and the only real variable is the lifetime of a cookie after the last action. The lifetime must be value which will be accepted by CGI::Session. The directory contains the management informations for the cookies (like the name of the user).

Example 4.4. Session configuration

```
<session>
    <type>cookie</type>
    <directory>@var_prefix@/session/cookie</directory>
    <lifetime>1000</lifetime>
<session>
```

1.4. ACLs

The basic configuration of OpenCA's access control list is placed in OPENCADIR/etc/access_control/*.xml. The relevant datastructure is like follows:

Example 4.5. Basic ACL configuration

```
<acl_config>
    <acl>yes</acl>
    <list>@etc_prefix@/rbac/acl.xml</list>
    <command_dir>@etc_prefix@/rbac/cmds</command_dir>
    <module_id>0</module_id>
    <ca_cert>@var_prefix@/crypto/cacerts/cacert.pem</ca_cert>
    <map_role>no</map_role>
    <map_operation>no</map_operation>
</acl_config>
```

Following you can find the meanings of the elements:

acl	enable (yes) or disable (no) the access control list
list	defines the place of the ACL
command_dir	specify the directory which contains the configuration files of the scripts
module_id	This is the id of the interface. This id is unique for every interface.
ca_cert	The CA certificate in PEM format is used to verify signatures. You can use here another certificate than of the CA which you control with this access control. This is useful if you have a user CA and a server CA. You can login into the interface of the server CA with a certificate from the user CA.
map_role	enable (yes) or disable (no) the mapping from certificates to roles if the certificate of the user is known
map_operation	enable (yes) or disable (no) the mapping from the names of the scripts to the supplied operation

You should check the access control list itself very carefully after you initialized the basic configuration of the interface. The real access control list is embedded into the access control area and has the following format:

Example 4.6. Permission for serverInfo

```
<acl>
    <permission>
        <module>0<</module>
        <role>root<</root>
        <operation>serverInfo</operation>
        <owner></owner>
    </permission>
    <permission>...</permission>
    ...
</acl>
```

The meanings of the elements are the following ones:

permission	every access right is defined in a permission element
module	this is the module id of the used web interface. Every installed gateway of OpenCA is a module in the terms of RBAC. If you install the RA then there is a new module with the id `1`.
role	is the name of the user if `map_role` is *`no`* and the role of the certificate if `map_role` is *`yes`*. The roles are part of every role based system. OpenCA defines a set of default roles which you can simply extend by other roles which you need. Every certificate will be assigned a role if it is issued on the CA. The role of a certificate service request is the role which the requests asks for. The role of a certificate revocation request is the role of the certificate to which the CRR belongs. The CA-certificate(s) and the CRLs have no explicit role because they have automatically the “superrole”. If there is an action where the user is not identified by a certificate then the role which is used is automatically the empty role. This is sometimes necessary for example if you want to control your public gateway by RBAC.
operation	is the name of the script if `map_operation` is *`no`* and the operation of the script if `map_role` is *`yes`*.
owner	is the role of the object which will be automatically detected by the configuration of the script

All options can be used with regular expressions but of course the parameters are case sensitive. An ACL which allows anything and only requires a valid login looks like this:

Example 4.7. Allow all

```
<acl>
    <permission>
        <module>.*</module>
        <role>.*</role>
        <operation>.*</operation>
        <owner>.*</owner>
    </permission>
</acl>
```

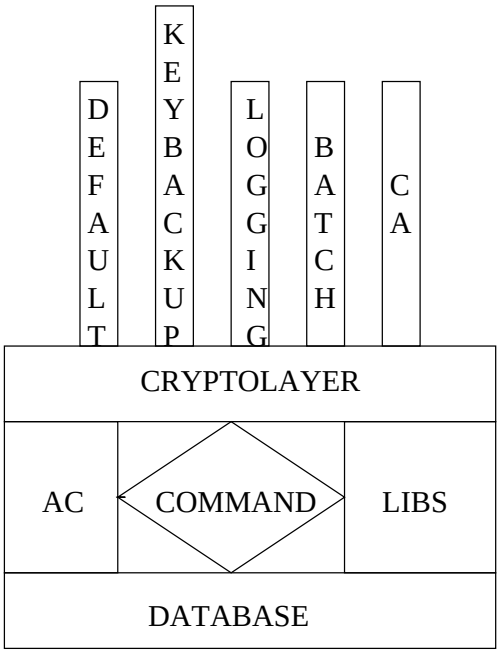
Every command or script has it's own configuration file which contains the name of the command (this is actually a protection against the renaming of files), the name of the operation for which it is used, the way how to find the affected object and the name of the variable which contains the data which is necessary to determine the object by the specified way. Today there are six `OWNER_METHOD`s:

`CERTIFICATE_SERIAL`	This method is used if an operation affects a certificate and the role should be detected by the serial of the certificate.
`REQUEST_SERIAL`	This method is used if an operation affects a CSR and the role should be detected by the serial of the CSR.
`CRR_SERIAL`	This method is used if an operation affects a CRR and the role should be detected by the serial of the CRR. The CRR will be loaded and the certificate which should be revoked will be loaded and the role of the certificate is used.
`CGI`	The use of this method is not recommended because the role is not protected by any cryptographic mechanisms.
`ANY`	The operator must have the right to perform this operation for every role.
`<empty>`	This method is used to signal that an object is handled which affects the CA directly (e.g. CA-certificate, CRL). The operator needs access to the “superrole”.

2. Token and keyconfiguration

OpenCA has a concept which abstracts every key. Every key is a token and be putted into a slot. This mean that the software can ask the cryptolayer for a token `ca` and the cryptolayer checks it's configuration for a token called `ca`.

Figure 4.3. Tokenconcept



Now we will explain the token configuration because this is the most interesting thing for an administrator. The basic schema is the following one:

```
<openca>
    <token_config>
        <default_token>CA</default_token>
        <token>...</token>
        ...
    </token_config>
</openca>
```

Every token configuration includes some common parts which will be described here. The names of the options are tag names!

name — which will be used by the software for the token. There are four names today - `CA`, `BP` (batch processors), `KEYBACKUP` and `LOG`.

type — This defines the type of the token. Today there are three types (`OpenSSL`, `Empty` and `LunaCA3`) but we can add modules for every token you need if the token is supported by OpenSSL. We add a module per token to be able to handle the different details and proprietary tools to manage the tokens.

mode — The mode describes how the token should be used. OpenCA knows three modes `standby`, `session` and `daemon`. `standby` means that you must login to the token for every action, `session` activates the token for the whole user session and `daemon` runs the token in a daemon mode which means that the token must be stopped explicitly. Please read the docs about the different token types because not every token supports every mode.

login_sheet — If the token login requires a password which the user have to enter at the webpage then you should specify a sheet where the user can do it. Usually there are prepared sheets already installed at `OPENCADIR/lib/servers/server_name/sheets/token_login.html`. So you have only to specify the default sheet.

We will describe now the configuration of different token types.

2.1. OpenSSL

OpenSSL only support the operational mode *`standby`*. Additionaly it support several other dynamic options which are required to work properly.

SHELL — This is the path to binary of OpenSSL. It is called SHELL because it is the cryptoshell which we use. Simply run the binary without any options and you know what we mean.

KEY — filename of the file with the private key

PASSWD_PARTS — the number of the components of the passphrase. If you have two groups of administrators and every group has only one part of the passphrase then you can enter *2* and OpenCA will display two seperate input fields for the different parts.

PEM_CERT — path to the PEM formatted certificate

DER_CERT — path to the DER formatted certificate

TXT_CERT — path to the TXT formatted certificate - this is only important for the CA token.

CHAIN — directory which includes the certification path - this is not only important for the CA because the chain is sometimes included into signatures.

2.2. Empty

This token is only a dummy if the key is not a really seperate key. Often the administrators simply use symlinks to the CA keys and certs for the keybackup etc.. An empty token is redirect to the default token which is usually the CA token.

2.3. LunaCA3

This token accepts all the options like OpenSSL tokens except of `PASSWD_PARTS` because Chrysalis-ITS Luna CA3 uses hardware (PIN pad) for login to prevent electronic reconnaisance actions. Luna CA3 supports all operational modes (*`standby`*, *`session`* and *`daemon`*. This module requires some additional options:

UTILITY — Luna CA3 comes with a utility to manage things like login and keygeneration. You have to enter the complete path here.

SLOT — This is an ID for the slot of the token.

APPID — This is the application ID to avoid conflicts with other application. Please remeber that the *`APPID`* must be lesser than the *`SLOT`*.

LOCK_FILE — OpenCA uses this file to detect that the token is already activated if the token runs in daemon mode. There is no way to find out this fact directly via a tool from Chrysalis-ITS.

3. OpenSSL

You must care about three configurationfiles and -directories `etc/openssl/openssl.cnf`, `etc/openssl/openssl` and `etc/openssl/extfiles`. The first file contains the configuration for the CA. This means the file is used for the generation of the initial CA-CSR, the selfsigned certificate (if you setup a Root CA) and the CRLs. The file is configured fullautomatically during the installation but if you are setting up a serious CA then you should check this file too. The directory `etc/openssl/openssl` contains the configuration for the different roles except of the extensions. The relevant things which you must compare with your policy are the lifetime of the certificate and the algorithm which is used to sign the certs. The dircetory `etc/openssl/extfiles` contains the definitions of the extension. Please check these files carefully.

3.1. Certificate Extensions

3.1.1. Standard Extensions

3.1.1.1. Authority Key Identifier

[[RFC3280] The authority key identifier extension provides a means of identifying the public key corresponding to the private key used to sign the certificate. This extension is used where an issuer has multiple signing keys (either due to multiple concurrent key pairs or due to changeover).]

The authority key identifier is used for path construction. Only if you create a self-signed root CA certificate then you only need the subject key identifier. You can use the subject key identifier of the CA certificate or/and the issuer's certificate serial and issuer name. It is recommended to use both. It is important to understand that the name is the name of the issuer of the CA certificate. If you have a root CA, a sub CA and a user certificate then the name in the authority key identifier is the subject of the root CA's certificate.

The value should look like this:

Example 4.8. OpenSSL configuration - Authority Key Identifier

```
authorityKeyIdentifier=keyid,issuer:always
```

Never mark this extension as critical.

3.1.1.2. Subject Key Identifier

3.1.1.3. Key Usage

3.1.1.4. Private Key Usage Period

3.1.1.5. Certificate Policies

3.1.1.6. Policy Mappings

3.1.1.7. Subject Alternative Name

3.1.1.8. Issuer Alternative Name

3.1.1.9. Subject Directory Attributes

3.1.1.10. Basic Constraints

3.1.1.11. Name Constraints

3.1.1.12. Policy Constraints

3.1.1.13. Extended Key Usage

3.1.1.14. CRL Distribution Points

3.1.1.15. Inhibit Any-Policy

3.1.1.16. Freshest CRL

3.1.2. Internet Certificate Extensions

3.1.2.1. Authority Information Access

3.1.2.2. Subject Information Access

3.1.3. Vendor Specific Extensions

3.1.3.1. Microsoft

3.1.3.2. Netscape

3.2. Profiles

3.2.1. HTTPS server

The different names of HTTPS servers are one of the most problematic things in the todays world. Like for many other cryptographic issues in the web there is a standard for servercertificates - RFC 2818 “HTTP over TLS”.

The standard defines that you have to check the subject alternative name for an appropriate entry (DNS or IP see RFC 2459). If this search fails then check the common name in the distinguished name of the certificate.

If you use Microsofts Internet Explorer then you have no problems. The IE is full standard compliant. The problem is Netscape. They use the common name like a regular expression in Unix. The common name can be in the format “(server1|server2).my_domain.org”. The clever ones would argue now that we must simply set the subject alternative name like defined by RFC 2818 and set a normal common name because the subject alternative is checked first. This is a nice idea but Netscape ignores the subject alternative name if it checks the name of the server versus the content of the certificate.

The solution is a mix between RFC 2818 and Netscape behaviour. You must set the common name in the distinguished name like Netscape defines it. After this you must set all DNS-names and the IPs of the server in the subject alternative name. If you do this then all standard compliant browsers will evaluate the subject alternative name first and will ignore the common name in the distinguished name. So the certificate is standard compliant but supports the cruel behaviour of Netscape too.

Before you think this is a perfect solution then please think about aliases. My personal record are 20 different names for one computer which I have to code in a common name. If you think that it is easy then please remember that a common name can only be 64 characters long.

3.2.2. SMTP server

Mailservers usually include SMTP daemons. SMTP servers act as server and client because they work in a hierarchy. Some server softwares like sendmail require that the SMTP server identifies itself as a SMTP client if it contacts another SMTP server. Usually you only want to issue one certificate per server and not one certificate per service and therefore you have to set the extensions for SSL Client and SSL Server like recommended by OpenSSL (please see "man x509" after you installed OpenSSL). If you use sendmail then you can create a server certificate for the SMTP server and an additional client certificate. Sendmail supports two certificates per server or better per daemon.

SSL Client requires the following extensions:

Example 4.9. Minimal SSL client extensions

```
keyUsage = digitalSignature
extendedKeyUsage = clientAuth
nsCertType = client
```

SSL Server requires the following extensions:

Example 4.10. Minimal SSL server extensions

```
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, msSGC, nsSGC
nsCertType = server
```

msSGC and nsSGC mean Server Gated Crypto from Microsoft and Netscape.

If you want to configure only one certificate per SMTP server then you have to issue certificates which looks like a christmas tree. They have to include all extensions for clients and servers. A configuration can look like this:

Example 4.11. Minimal SMTP extensions for a single certificate

```
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, serverAuth, msSGC, nsSGC
nsCertType = client, server
```

3.2.3. F-Secure VPN+

If you want to use OpenCA with F-Secure VPN+ then you should bear in mind that this software can only download a CRL via http or ldap. They don't support https and ldaps. This is important if you configure your CRLDistributionPoints in `OPENCADIR/etc/openssl/extfiles/*.ext`. You can easily fix this problem by using LDAP for CRL-distribution.

Certificates for VPN+ Gateways and Machine certificates should include the DNS name and IP address in the subject alternative name. The certificates for the persons to authenticate them can be contain anything you want. It must only be valid client certificates.

4. CSRs

First we describe the concept of additional attributes and then we describe the two general types of requests - external prepared PKCS#10 requests and during the interaction generated requests.

4.1. Additional Attributes

Usually the first question is about what does an additional attribute be? Additional attributes where introduced to store extra informations in a request without publishing these informations. Big organizations or trustcenters need a lot of information and only a minimal subset should be public. If you are a postmaster or a webmaster of a university then it is a good idea to put the general emailaddresses into the certificates but it is not really optimal to store the telephonenumber in the certificate. Nevertheless it makes sense for the trustcenter stuff to have the phonenumber in a case of emergency.

After you know what a nice feature these attributes are you want to customize them? No problem. A simple example should help:

Example 4.12. Additional attributes configuration

```
ADDITIONAL_REQUEST_ATTRIBUTES   "requestercn" "email" "department" "telephone"
ADDITIONAL_ATTRIBUTES_DISPLAY_VALUE     "Name (first and Last name)" "Email" "Department"
ADDITIONAL_REQUEST_ATTRIBUTES_STRING_TYPE "LATIN1_LETTERS" "EMAIL" "LATIN1_LETTERS" "LATIN
```

The three options have the following meanings:

ADDITIONAL_REQUEST_ATTRIBUTES	These are the internal names for the attributes.
ADDITIONAL_ATTRIBUTES_DISPLAY_VALUE	Here you have to define the displayed names for the attributes. This is useful for example if you have two public interfaces for example an english and a german one.

ADDITIONAL_REQUEST_ATTRIBUTES_STRING_TYPE

This helps the software with the errordetection. You can use the following values:

- LETTERS
- TEXT
- NUMERIC
- MIXED
- DATE
- TEL
- EMAIL
- LATIN1_LETTERS
- LATIN1

4.2. PKCS#10 Requests

The first certificates which were needed in the open source are were server certificates. The most systems which use such certificates are Apaches, OpenLDAPs, IMAPDs, POPDs, SMTPDs and S-Tunnel. Such systems generate the private key by itself - means the administrator generate a key by hand or via the software but there is no interaction with the trustcenter's software until the administrator created a request with the key or exported a request from the software.

If the administrator has the PKCS#10 request then he must bring the request to the trustcenter and this is the job of command which's configuration we describe here. The option `DN_TYPE_PKCS10_REQUIRED_ELEMENTS` define the structure of the subject of the PKCS#10 request. The option `DN_TYPE_PKCS10_BASE` and the values of `DN_TYPE_PKCS10_BASE_*` define the suffix for the certificates which will be accepted by this interface. The restrictions were implemented to get some kind of useful subjects.

Example 4.13. PKCS#10 configuration

```
DN_TYPE_PKCS10_REQUIRED_ELEMENTS "CN" "OU" "O" "C"
DN_TYPE_PKCS10_BASE     "O" "C"

DN_TYPE_PKCS10_BASE_1 "OpenCA"
DN_TYPE_PKCS10_BASE_2 "it"
```

4.3. Basic CSR

If you have no prepared PKCS#10 request then there is a second method in OpenCA. This method is used if the key and request generation happen during the interaction with the client. OpenCA support clientside keygeneration for Microsoft Internet Explorer, Netscape Communicator, Mozilla and Opera. If you have another browser then OpenCA uses it's serverside key and requestgeneration. So let's start with an example:

Example 4.14. Basic CSR configuration

```
Basic_CSR_Keysizes "512" "768" "1024" "2048" "4096"

DN_TYPES "BASIC"
DN_TYPE_BASIC_KEYGEN_MODE "SERVER"
DN_TYPE_BASIC_KEYGEN_SHEET "@lib_prefix@/servers/@pub_prefix@/sheets/basic_csr_confirm

DN_TYPE_BASIC_BODY "Y"
DN_TYPE_BASIC_BASE "O" "C"
DN_TYPE_BASIC_ELEMENTS "emailAddress" "CN" "OU"
DN_TYPE_BASIC_NAME "Basic User Request"

DN_TYPE_BASIC_BASE_1 "@ca_organization@"
DN_TYPE_BASIC_BASE_2 "@ca_country@"

DN_TYPE_BASIC_ELEMENT_1                "E-Mail"
DN_TYPE_BASIC_ELEMENT_1_MINIMUM_LENGTH 7
DN_TYPE_BASIC_ELEMENT_1_REQUIRED       "YES"

DN_TYPE_BASIC_ELEMENT_2                "Name"
DN_TYPE_BASIC_ELEMENT_2_MINIMUM_LENGTH 3
DN_TYPE_BASIC_ELEMENT_2_REQUIRED       "YES"

DN_TYPE_BASIC_ELEMENT_3                "Certificate Request Group"
DN_TYPE_BASIC_ELEMENT_3_SELECT         "Internet" "Partners" "Employees" "Trustcenter"
DN_TYPE_BASIC_ELEMENT_3_MINIMUM_LENGTH 8
DN_TYPE_BASIC_ELEMENT_3_REQUIRED       "YES"
```

The first line defines the available keysize. The next variable DN_TYPES defines the available configurations of **basic_csr**. The command **basic_csr** is called via a link and the link must contain an option CSR_TYPE which defines the configuration which is used for this CSR. If you don't set this option then **basic_csr** starts it's browserdetection.

The default type which is supported by OpenCA is *BASIC*. You can simply add a type and set a correct link in the public gateway. You can find an example on the public gateway by looking at the link “Basic Request”.

The prefix of every definition is now DN_TYPE_BASIC_. The NAME defines the displayed name (e.g. "Request for managers only"). The BODY defines the type of the request. If the value is *Y* or *YES* then a key and a request will be stored and if necessary generated. If the value is *N* or *samp* then only a header will be generated. This option is used to get the necessary data from a user to initialize a smartcard on the registration authority.

DN_TYPE_BASIC_KEYGEN_MODE specifies the way how to generate a key and request. The supported modes are SERVER, SPKAC and IE. SPKAC is used with Opera, Mozilla and Netscape, IE is used for Microsoft Internet Explorer and SERVER is used for all other situations.

The BASE is the part of the subject which is not editable by the user who requests a certificate. The other BASE_numbers define the values of the elements which are used for the not editable part of the subject.

The ELEMENTS are the part of subject which can be defined by the user. The defined attributes of the subject can be configured more precisely by the options named *_ELEMENT_number*. They have the following meaning:

*_ELEMENT_number — These are the displayed names of the elements. The normal user don't know what is a CN or a commonName. The most users will be confused if they see two fields with the

same name (e.g. OU). So you can give the attributes some names which match their semantic.

`*_ELEMENT_number_MINIMUM_LENGTH` — This field defines what the minimum length of the value is. If you don't know it then simply use *0*.

`*_ELEMENT_number_REQUIRED` — Usually the user has to fill all fields but sometimes it is a good idea to have some optional fields. If you have such an optional field then please set the value to something different than *"YES"*. If a value is entered by the user then the option `*_ELEMENT_number_MINIMUM_LENGTH` still will be checked.

`*_ELEMENT_number_SELECT` — All fields are textfields by default. You can specify `*_ELEMENT_number_SELECT` followed by a list of values. OpenCA creates a HTML-select from this definition.

`*_ELEMENT_number_XML_FILE` — If you need some more options or you have an export from an ERP database then there is an additional method to create HTML-select fields. You can create a XML file which must contain a list. The filename you must specify here.

`*_ELEMENT_number_XML_PATH` — If you specified an XML file then you have to specify the XPath to this list too. The XPath for the following example is "basic_csr/basic/element_3/option":

Example 4.15. Configuration example for a XML file based HTML-select

```
<openca>
    <basic_csr>
       <basic>
           <element_3>
               <option>Computer staff</option>
               <option>Management</option>
               ...
           </element_3>
       </basic>
    </basic_csr>
</openca>
```

4.4. SCEP

OpenCA supports SCEP for sending requests but we define no rules for such requests because the clients are not able to interact with an interface and so we accept every request which arrives.

5. Subject

5.1. Common stuff

OpenCA displays at every time DNs like defined by RFC 2253. There are five options which influence the subject during the issuing itself:

SET_REQUEST_SERIAL_IN_DN	This options enforce the inclusion of the request's serial in the subject of the certificate. This is a simple method to guarantee that the subject is unique. True values are *Y*, *YES* and *ON*.
REQUEST_SERIAL_NAME	If the serial of the request will be included then this option defines which attribute is used for the serial.
SET_CERTIFICATE_SERIAL_IN_DN	This options enforce the inclusion of the certificate's serial in the subject of the certificate. This is a simple method to guarantee that the subject is unique. This option is more recommended than `SET_REQUEST_SERIAL_IN_DN` because the value is tranparent. True values are *Y*, *YES* and *ON*.
CERTIFICATE_SERIAL_NAME	If the serial of the certificate will be included then this option defines which attribute is used for the serial.
DN_WITHOUT_EMAIL	This option is used to enforce recommendations of S/MIME v3. If you don't want to include the emailaddress in the subject then you can use this option. OpenCA will remove the emailaddress from the subject before it issues the certificate. True values are again *Y*, *YES* and *ON*.

5.2. dc style

OpenCA uses by default the old “o=University,c=de” style. Several users like international companies, universities or other big organizations need the new dc style. Therefore we support the dc style too. It is necessary to change several files because the configuration of the subjects is highly integrated into the software. We will explain it with an example.

```
base dn or suffix: dc=university,dc=edu
user dn: dc=mike tester,dc=university,dc=edu
webserver dn:dc=www,dc=university,dc=edu
ca dn:dc=CA,dc=university,dc=edu
```

There are five things which you have to check for the change to the dc style. The steps will be now described:

5.2.1. etc/servers/*.conf

There are two things which must be changed in the configuration files of the servers.

The LDAP configuration must be adapted to the new dc-style. The variables - which you must modify - are `basedn` and `ldaproot`. The `basedn` is the suffix of the LDAP server. The `ldaproot` is the dn of the user root to bind to the LDAP server. The `ldaproot` has not to be changed because it is freely configurable by the administrator of the LDAP server.

```
basedn "dc=university,dc=edu"
ldaproot "dc=manager,dc=univesity,dc=edu"
```

The configuration of the requests must be changed too because they are prepared for the old style. Please read the following example to get an overview of a dc-styled configuration. Please read the section about the CSR configuration to understand how the normal requests can be configured.

```
DN_TYPE_BASIC_BODY "YES"
```

```
DN_TYPE_BASIC_KEYGEN_MODE "SERVER"
DN_TYPE_BASIC_KEYGEN_SHEET "/usr/local/OpenCA/lib/servers/pub/sheets/basic_csr_confirm_req

DN_TYPE_BASIC_BASE "DC" "DC"
DN_TYPE_BASIC_ELEMENTS "DC"

DN_TYPE_BASIC_NAME "Basic User Request"

DN_TYPE_BASIC_BASE_1 "University"
DN_TYPE_BASIC_BASE_2 "edu"

DN_TYPE_BASIC_ELEMENT_1 "Name"
```

5.2.2. main.html

Please check the installed or prepared files with the name `main.html` because several HTML files display the suffix of all the DNs.

5.2.3. certsMail.txt and expiringMail.txt

You can find these files in `lib/servers/ra/mails`. They are the default templates for the mails which RA Operators can send to the users. They include the suffix of the LDAP server. This suffix is called `Dir Root`. This suffix must be changed according to the real suffix of your LDAP server.

5.2.4. OpenSSL configuration

You must modify the files `OPENCADIR/etc/openssl/openssl.cnf` and `OPENCADIR/etc/openssl/openssl/*.conf`. The policy and req sections must be changed to support requests and certificates with subjects in the dc-style. If you don't know how to configure OpenSSL then please read the documentation of OpenSSL.

5.2.5. CA CSR

If you generate the initial request for the CA request then please ignore all the fields for the normal subjectstyle. Simply enter nothing in all field until the software displays the window which show you the complete subject. There you have to enter the complete subject of the CA request. The subject is in RFC 2259 format and all “DC” must be written in big letters because OpenSSL is case sensitive.

6. Subject Alternative Name

The most people want to use OpenCA for issuing certificates to users or they want to test a PKI without buying a commercial trustcenter only for testing. So they want create a request, approve the request and issue a certificate. The problem is that they forget to the edit the request and the subjct alternative name was not set.

OpenCA knows two switches in `ca.conf` to set the subject alternative name automatically. The switch `AUTOMATIC_SUBJECT_ALT_NAME` enables the mechanism to set the subject alternative name automatically if it was not set in the header of the request. The second switch `DEFAULT_SUBJECT_ALT_NAME` defines the type of the default value. Actually we implemented only support for the emailaddress. If you need support for DNS name(s) or IP addresse(s) then contact us. We only don't implement it because nobody need it until now.

If you edit the subject alternative name on the RA interface then you see only four fields where you can enter parts of the alternative name. If you fill all fields then you will get at next time you want to edit the alternative name one addi-

tional field. If you know that you need more or less fields then you can change the option `CSR_MAX_SUBJECT_ALT_NAMES` in the configuration files. The option defines the number of the displayed fields by default. It is NOT a hard limit.

7. LDAP

OpenCA provides an LDAP interface for users to download certificates from a central repository. This interface can be utilised by browser address books and specialised LDAP clients.

Before the OpenCA Online components can write certificates and CRLs to the directory you must have an LDAP compliant directory installed and available to the online components (this can be on the same or different machine). One example of an appropriate directory is the OpenLDAP project.

7.1. Configuration of the Directory

A full description of the configuration of your LDAP directory is outside the scope of this document. Important points to note are:

- Ensure that the following schemas are included (probably in the slapd.conf file):
  - core.schema
  - cosine.schema
  - inetorgperson.schema
  - openca.schema
- Ensure the directory is started with the appropriate suffix (e.g. o=myorg,c=gb).
- Ensure the rootdn is specified.
- Ensure the root password is specified.

7.2. Configuration of the online components

Three configuration files must be configured for the online components to make use of the LDAP directory to store certificates; `OPENCADIR/etc/servers/node.conf`, `OPENCADIR/etc/servers/ldap.conf` and `OPENCADIR/etc/ldap.xml`. If you don't understand the configuration options then please read in the technology guide for more informations. Usually it is enough to set the correct options in `ldap.xml` or in `config.xml`.

7.3. Writing Certifciates to the Directory

As long as the option `updateLDAPautomatic` is set to *`yes`* the online components will attempt to upload certificates to the directory after an import. Before this can happen the directory must be initialised and the appropriate structure must be implemented. In this version of OpenCA this initialization is done automatically.

8. SCEP

SCEP is the successor of CEP the Certificate Enrollment Protocol. Both protocols were developed from Cisco. The idea is to have simple but secure protocol to enroll certificates and CRLs. Today many network components use SCEP to manage certificates and CRLs. Some of these components are Switches, Routers, Firewalls and VPN-Softwares.

OpenCA support SCEP via an own web interface. The interface is called scep and you can install it via "make install-scep". After the installation you have only to configure the file `OPENCADIR/etc/servers/scep.conf` or you edit `OPENCADIR/etc/config.xml` before you run **OPENCADIR/etc/configure_etc.sh**. Please remember to only filter via IP addresses because SCEP doesn't support any authentication mechanisms. A SCEP client can connect the interface via *`http://your_host/cgi-bin/scep/scep`*.

8.1. `OPENCADIR/etc/servers/scep.conf`

This file contains the followin parameters:

ScepRAKey	This is the PEM encoded private key of the SCEP interface. It has the same format like for mod_ssl.
ScepRACert	This is the PEM encoded certificate of the SCEP interface. It has the same format like for mod_ssl.
ScepRAPasswd	This is the passphrase for the private key of the SCEP server. If you use a not encrypted private key (what is not recommended - then please set an empty string here. interface. It has the same format like for mod_ssl.

8.2. `OPENCADIR/etc/config.xml`

This file contains the followin parameters:

SCEP_RA_KEY	This is the PEM encoded private key of the SCEP interface. It has the same format like for mod_ssl.
SCEP_RA_CERT	This is the PEM encoded certificate of the SCEP interface. It has the same format like for mod_ssl.
SCEP_RA_PASSWD	This is the passphrase for the private key of the SCEP server. If you use a not encrypted private key (what is not recommended - then please set an empty string here. interface. It has the same format like for mod_ssl.

9. Dataexchange

9.1. Configuration

The dataexchange of OpenCA is highly configurable. So first we have to describe some general concepts.

If you look at OpenCA from a database viewpoint then OpenCA is a tree of hierarchical organized databases. Every database is used by some web interfaces. So one node of the hierarchy consists of a database and some web interfaces. If we describe the dataexchange then we describe the dataexchange between nodes. This is the reason why we called the management interface node.

A node can exchange data with a node of a higher level of the hierarchy or with several nodes which are on a lower level of the hierarchy. If export data to a higher level of the hierarchy then we UPLOAD data and if we import data from such a node then we DOWNLOAD data. If import data from a lower level then we RECEIVE data and if export data to such a node then we ENROLL data.

If you exchange object in a security relevant area then you must define which object with which state you want to exchange. Therefore you can define in OpenCA which objects with which state you accept from which direction. Also OpenCA allows only to overwrite existing objects if you DOWNLOAD CA-certificates, CRLs, CSRs and CRRs. Status or object injections are not accepted in all other situations. OpenCA includes some default configurations to help you on the way to secure configuration.

The following example is for the import from a higher level of the hierarchy:

Example 4.16. Download configuration

```
DOWNLOAD_CA_CERTIFICATE_STATES VALID
DOWNLOAD_CERTIFICATE_STATES    VALID
DOWNLOAD_CRL_STATES            VALID
DOWNLOAD_CRR_STATES            ARCHIVED DELETED APPROVED
DOWNLOAD_CSR_STATES            ARCHIVED DELETED
DOWNLOAD_MAIL_STATES           CRINS DEFAULT
```

The export/import technology itself is another important aspect. You can configure OpenCA to use different methods for the dataexchange with higher and lower levels of the hierarchy. Therefore we implemented options to prepare and cleanup IO operations. This is necessary if you have to start special networkinterfaces or to mount devices. There are also options to configure the EXPORT and IMPORT of the directory with the data and there is an option to TEST the result. EXPORT, IMPORT, START and STOP accept more then one argument. Every argument will be seperately executed. The parameters *@__SRC__@* and *@__DEST__@* include the directories with the data. The parameter *@__DEVICE__@* will be replaced with the value of DEVICE.

Example 4.17. Export configuration

```
EXPORT_IMPORT_UP_DEVICE "/dev/fd0"
EXPORT_IMPORT_UP_START ""
EXPORT_IMPORT_UP_STOP ""
EXPORT_IMPORT_UP_EXPORT "/bin/tar -cvfp @__DEVICE__@ -C @__SRC__@"
EXPORT_IMPORT_UP_IMPORT "/bin/tar -xvf @__DEVICE__@ -C @__DEST__@"
EXPORT_IMPORT_UP_TEST "/bin/tar -tvf @__DEVICE__@"
```

The hierarchylevel LOCAL is used for backups, batchprocessors and such things. It is also possible to configure OpenCA for the use with temporary private networks. Here is another example for a CA:

Example 4.18. Local export configuration

```
    EXPORT_IMPORT_DOWN_DEVICE "openca.tar"
    EXPORT_IMPORT_DOWN_START "/sbin/ifconfig eth0 up"
    EXPORT_IMPORT_DOWN_STOP "/sbin/ifconfig eth0 down"
    EXPORT_IMPORT_DOWN_EXPORT "/bin/tar -cvfp /usr/local/openca/var/tmp/@__DEVICE__@ -C @_
    EXPORT_IMPORT_DOWN_IMPORT "/usr/bin/scp openca@ra.openca.org:/usr/local/OpenCA/var/tmp
    EXPORT_IMPORT_DOWN_TEST ""
```

9.1.1. Configuration with simple files

The standard data exchange between RA and CA is done via `/dev/fd0` (floppy). If the CA and RA are really big then it is recommended to change the data exchange location to a simple file on the local system (for example `OPENCADIR/var/tmp/fd0`). Please always remember that your apache must be able to access this file. You can also do a chown and set the owner to apache's user.

You also have to edit the dataexchange sections of the .conf files in `OPENCADIR/etc/servers/` to point to the new file (edit the lines `EXPORT_IMPORT_*_DEVICE`).

It is recommended to erase the file after transfers, because the exchange can contain private keys of users.

9.1.2. Configuration via scp

If the CA and RA are located on different machines in a secure environment with perhaps an offline CA it can be difficult to do the dataexchange via simple files (the files have to be transferred between CA and RA, either via a diskette or ftp (not secure)). OpenCA offers the possibility to do the dataexchange automatically via scp.

The RA exports resp. imports from a file located on the system it is installed on (same as local datatransfer). Thus the RA configuration is the same as for the local dataexchange:

- Create a new file for the export with the correct permissions like `OPENCAIDR/var/tmp/fd0`
- Edit the ra_node.conf file in `OPENCADIR/etc/servers/` to point to the new file (edit the lines `EXPORT_IMPORT_*_DEVICE`).

To use scp you must have a working openssh environment with an ssh client on the CA side and an ssh server on the RA side. On the CA side determine the apache home directory by looking at the etc/passwd file (in redhat 8 and 9 this is /var/ww). Do the following:

SSH configuration

```
# On the CA side do the following:

$> mkdir /var/www/.ssh
$> cd /var/www/.ssh

# Now we generate a new private public key pair for the ssh connection
# between CA and RA. When prompted, name the key id_rsa_1024_openca.

$> sshkeygen -b 1024 -t rsa
$> cp id_rsa_1024_openca identity
$> chown -R apache:apache /var/www/.ssh

# Copy the id_rsa_1024_openca.pub public key to the RA.
# On the RA side do the following:

# Add a new user called openca:

$> adduser openca

# Change the password of this user to whatever you want.

$> passwd openca
$> mkdir /home/openca/.ssh
$> cp id_rsa_1024_openca.pub /home/openca/.ssh
$> cp id_rsa_1024_openca.pub /home/openca/.ssh/authorized_keys
$> chown -R openca:openca  /home/openca/.ssh
```

Now go back to the CA side to edit the files `ca.conf` and `ca_node.conf`. The `ca_node.conf` file already contains a sample scp datexchange section. Comment out the dataexchange section used for local datexchange. Uncomment the scp part and edit it:

- The `EXPORT_IMPORT_DOWN_DEVICE` is the file used for datexchange and must have the same name as the one defined in `ra_node.conf` on the RA side.
- The `EXPORT_IMPORT_DOWN_START` can have no value or a value like "ifconfig eth0 up" in which case the CA goes online for the dataexchange.
- The `EXPORT_IMPORT_DOWN_STOP` can have no value or a value like "ifconfig eth0 down" in which case the CA goes offline after the dataexchange.
- In the `EXPORT_IMPORT_DOWN_EXPORT` line we can define the operations OpenCA has to perform when doing an export (CA to RA transfer).
- In the `EXPORT_IMPORT_DOWN_IMPORT` line we can define the operations OpenCA has to perform when doing an import (RA to CA transfer).
- In the `EXPORT_IMPORT_DOWN_TEST` line we can define the operations that have to be performed to test the transfer (was it successful or not).

9.2. Adding a new node

If you create a new node e.g. a second RA then you have to support this node with the dataexchange mechanism. Every interface of OpenCA must have a unique module ID. OpenCA manage the complete dataexchange with the ID of the node interface. The node interface knows which object of which datatype was already received by another node.

If you want to create a new node then you must create the corresponding files in `OPENCADIR/var/log`. You have simply to create some files in the directories `OPENCADIR/var/log/enroll` and `OPENCADIR/var/log/download` depending on the direction which you use for epxort. These directories contain some files of the style `$number_$datatype`. $number is the module ID of the node to which you want to export the data. The datatype is from the exported objects.

If you created a new module ID (e.g. you setup another RA) then you have simply to touch the file `$number_$datatype`. The new file is empty and so all objects will be exported.

10. Databases

10.1. PostgreSQL

This is only a short introduction to PostgreSQL. This section should not replace the reading of the PostgreSQL Administration Guide but after the lecture of this section you should be able to setup OpenCA very easily with a real datbase and not only with DBM files. This is important because PostgreSQL has transaction support which is essential for serious database applications.

If we talk about real world databases then it is a good idea to mention pgadmin III. The old versions pgadmin I and II only support windows. The third version supports Unix too. So there is now a GUI like for MySQL available which can be used to manage the database. There are people who think that GUIs are not necessary but they enhance the management of several things a lot because not everybody has the time and skill to optimize it's PostgreSQL database perfectly.

Another note about security, please never trust database security mechanisms fully. Use at every time a small IP-firewall in front of your database server. The default installation of OpenCA is a datbase server on localhost. If you want to install a database server on a different machines than the OpenCA components then always install these servers at minimum in a DMZ. Databases like PostgreSQL has today it's own strong security mechanisms but they have from time to time some big bugs. OS-based IP-filters like ipf or netfilter are usually more robust.

10.1.1. Basic Setup

After the installation of PostgreSQL there is usually a use postgres. Please use **su - postgres** to use this identity. After this step you should login into the database with psql. This is possible because the database template1 is always present on new databaseservers. The first real step is the creation of the user openca with **create user**. After this you logout and login again as the newly created user. Now you create the database openca. After this the database itself is ready for use but you should make the database a little bit more secure. Sometimes the database is already secure then you have problems to issue the psql commands. Please edit the `pg_hba.conf` first to get the access permissions for the openca user.

PostgreSQL Database and User Creation

```
su - postgres
psql -d template1
psql> create user openca with password '1234567890' createdb nocreateuser;
psql> \q
psql -d template1 -U openca
psql> create database openca;
psql> \q
```

You can restrict the database via PosgrSQL's own configuration. The directory `/etc/postgres` contains usually a file `pg_hba.conf` which controls the access to the database. There should only be one entry which looks like follows:

PostgreSQL Access Control pg_hba.conf

```
# TYPE    DATABASE  USER      IP_ADDRESS    MASK              USERAUTH  MAP
host      openca    openca    127.0.0.1     255.255.255.255   md5
```

This configuration restricts the access to the PostgreSQL database to the database openca via users from localhost. Nobodyelse can access the database now. Please remove all the other access rights - especially the lines which use IP-based trust settings. After the configuration you must restart PostgreSQL. The tables can be created with OpenCA's web interfaces.

Please ensure that the PostgreSQL server is connected to the network. Usually the TCP/IP socket is deactivated.

PostgreSQL TCP/IP in `postgresql.conf`

```
tcpip_socket = true
```

Sometimes the command of PostgreSQL changes. So please try to run the command **\h** in psql to see a list of the actual available commands.

10.1.2. Backup

PostgreSQL maintains three programs to dump a database - pg_dump, pg_dumpall and pg_dumplo. We describe pg_dump here because we don't like to backup all other databases and we don't like ot backup large objects only. The following options are used:

- large object are included
- create the database newly on recovery
- export all data as insert commands to support on the fly database mogration

- columnname based inserts to support rearrangement of column ordering
-

PostgreSQL Backup Command

```
## normal fully portable SQL export
pg_dump --file=backup.sql --format=p        \
        --create --inserts --column-inserts \
        -U openca openca

## tar-based SQL export for pg_restore
pg_dump --file=backup.tar --format=t        \
        --create --inserts --column-inserts \
        --blobs                             \
        -U openca openca

## custom compressed SQL export for pg_restore
pg_dump --file=backup.pgc --format=c        \
        --create --inserts --column-inserts \
        --blobs                             \
        -U openca openca
```

10.1.3. Recovery

If we have different backup methods then we have different recovery procedures too. The main difference is that we can use the plain text export directly with psql to import the database. This is possible because this file only includes pure SQL statement. The PostgreSQL specific exports must be processed with pg_restore because they include support for BLOBs too. There is only one command for both PostgreSQL formats because pg_restore detects the used format automatically.

PostgreSQL Recovery Command

```
## normal fully portable SQL export
psql -f backup.sql

## tar-based SQL export for pg_restore
## custom compressed SQL export for pg_restore
pg_restore --create --orig-order --verbose -U openca backup.(tar|pgc)
```

10.2. DBM Files

DBM file are much easier to handle than real SQL databases. If you want to use these database then you must only ensure that the directory which should contain the database files is fully writeable, readable and accessible by the webserver user. This will be handled by OpenCA's installation routine automatically. Sometimes in the past the users choose the wrong webserver user. The result is a message in the logs that the function configError doesn't exist. This

happens because the OpenCA script cannot load the library files. The new versions of OpenCA (0.9.1.4+) display correct errormessages in this case. Extra actions by the installing administrator are not necessary.

Please never forget that DBM files don't support transactions. If you implement a real world PKI then it is strongly recommend to use SQL databases to have a consistent state of the PKI even in the case of a system crash. DBM has also problems with multi user access for example on web servers with high loads.

10.2.1. Backup and Recovery

The advantage of DBM files is the use of plain files. You can simply use tar to backup and recover this database.

DBM Backup and Recovery Procedure

```
## create GNU backup
tar -czf dbm.tar.gz -C $OPENCADIR/var/ db/

## install GNU backup
tar -xzf dbm.tar.gz -C $OPENCADIR/var/

## create Unix backup
cd $OPENCADIR/var/
tar -cf dbm.tar db
gzip dbm.tar

## install Unix backup
cd $OPENCADIR/var/
gunzip dbm.tar.gz     ## gzip -d is identical
tar -xf dbm.tar
```

11. Email

11.1. Sendmail with basic SMTP authentication

The following example shows how to configure sendmail to send mails via a mailserver which requires a login. This basic authentication is today very common to protect the mailservices especially of big providers against spammers. The example was developed and tested with Microsoft Exchange but it should work with any other mailserver (e.g. sendmail or postfix).

Sendmail configuration for basic SMTP authentication

Example 4.19. `/etc/mail/sendmail.mc`

```
divert(-1)dnl
dnl # Example for establishing email distribution with sendmail(client) via Microsoft Exch
dnl # This is the sendmail macro config file for m4.
dnl # If you make changes to
dnl # /etc/mail/sendmail.mc, you will need to generate the
dnl # /etc/mail/sendmail.cf file (by means of the sendmail-cf and m4)

dnl # general: comments start with dnl, statements end with dnl
dnl # command for creating sendmail.cf:
dnl # =>  m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
dnl # requisite: sendmail-cf

dnl # include sendmail-cf path:
include(`/usr/share/sendmail-cf/m4/cf.m4')dnl
VERSIONID(`setup for Red Hat Linux')dnl
OSTYPE(`linux')dnl

dnl # initiating masquerading
GENERICS_DOMAIN(`localhost.localdomain localhost')dnl

dnl # create genericstable (file in /etc/mail)and insert records for name mapping:
dnl # root: first.last@name.com
dnl # openca: first1.last1@name1.com
dnl # initialising of genericstable
dnl # =>  sendmail -bi -oA/etc/mail/genericstable

dnl # activating masquerading
FEATURE(genericstable)dnl
FEATURE(masquerade_envelope)dnl

dnl # mailserver where we connect to
define(`SMART_HOST',`<URL mailserver>')dnl

FEATURE(redirect)dnl
FEATURE(nocanonify)dnl
define(`SMTP_MAILER_FLAGS',`e')dnl

dnl # some time-outs
define(`confTO_CONNECT', `1m')dnl
define(`confTO_IDENT', `0')dnl

dnl # since this mailserver requires logon authentication we have to
dnl # provide this information
dnl # create auth-info (file in /etc/mail), insert record:
dnl # AuthInfo:<URL mailserver> "U:<user>" "P:<password>" "M:LOGIN"
dnl # this mailserver requires AUTH LOGIN
dnl # sendmail requires db-file:
```

```
dnl # =>  makemap hash /etc/mail/auth-info.db <etc/mail/auth-info

FEATURE(`authinfo', `hash -o /etc/mail/auth-info.db')dnl
dnl # we use SMTP
MAILER(smtp)dnl

dnl # while testing:
dnl # changes in sendmail.mc: compile with m4 to new sendmail.cf
dnl # =>  service sendmail stop
dnl # =>  service sendmail start
dnl # or for testing start sendmail as demon and logging dialog to file:
dnl # => sendmail  -bd  -O LogLevel=14  -X /tmp/out1.log

dnl # for basic checking, set up manual communication with mailserver
dnl # => telnet <URL mailserver) 25
dnl # => ehlo localhost
dnl # ? <receiving basic info>
dnl # =>  AUTH LOGIN
dnl # mailserver replies b|ase64 encoded, we have to reply base64-encode
dnl # => <first user> ?
dnl # => <then password>?
dnl # => MAIL FROM: <valid mail-address>
dnl # => RCPT TO: <vaild mail-address>
dnl # => DATA
dnl # ?
dnl # stop communication with <new line> . <new line>
dnl # QUIT
```

Part III. User Guide

Table of Contents

Preface lviii
5. Interface Descriptions 59
1. Public PKI Server 59
1.1. CA 59
1.2. User 59
1.3. Certificates 61
1.4. Requests 62
2. Registration Authority 63
2.1. Administration 63
2.2. Pending Requests 63
2.3. Information 64
2.4. Utilites 64
3. Registration Authority Node 64
3.1. Gateways 64
3.2. Administration 65
3.3. Utilites 67
4. LDAP Interface 67
4.1. Update LDAP 67
4.2. View CA-Certificates 68
4.3. View Certificates 68
4.4. View CRLs 70
6. Functionality Descriptions 71
1. CA Initialization 71
1.1. Phase I: Initialize the Certification Authority 71
1.2. Phase II and III: Create the initial administrator and RA certificate 73
2. Node Initialization 75
3. CSR Handling - a request HOWTO 76
3.1. Ways to request a certificate 76
3.2. Edit a certificate signing requests 78
3.3. Approve certificate signing requests 79
3.4. Issue a certificate from a certificate signing request 79
3.5. Certificate enrollment 79
3.6. Delete certificate signing requests 79
4. Certificate Handling 79
4.1. Find a certificate 80
4.2. Download 80
4.3. Start revocation 81
4.4. Write an email to the owner 81
4.5. Informational messages and their meaning 81
5. SCEP 81
5.1. SSCEP 81
5.2. NetScreen ScreenOS 83
5.3. F-Secure VPN+ 83
7. Client Support 85
1. Introduction 85
2. Mozilla 85
2.1. General 86
2.2. Mozilla 87
2.3. Netscape 4 87
2.4. Opera 87
3. Microsoft 87

3.1. Internet Explorer 87
3.2. Outlook 87
3.3. Outlook Express 87

Preface

This is the ultimate user guide for OpenCA. Here we want to answer all the questions which user can have if they use OpenCA.

Chapter 5. Interface Descriptions

1. Public PKI Server

This section describes the public interface to the OpenCA PKI. From these screens a user can view current certificate lists, manage certificates and download CA and revocation certificates.

1.1. CA

This section describes the CA related utilities a user can access. Each heading below relates to a link on the left hand menu bar.

1.1.1. Get CA Certificate

By hitting this link the user is presented with a page titled "CA-Certificate Page". This page contains links to CA certificates in various formats.

In order for the user to "trust" certificates generated through OpenCA they must have the Certificate Authority root certificate installed. This page provides an easy mechanism for them to do that. Most users will just click the "CA-certificate in format CRT" and follow the instructions presented to them by their environment (e.g. In IE they have the option to "Open" the file and then "Install Certificate").

Apache Web server administrators would use the link "CA-certificate" to download the certificate in the appropriate format for inclusion in the Apache configuration files.

1.1.2. Certificate Revocation Lists

By hitting this link the user is presented with a screen entitled "CRL Page". This page contains links to certificate revocation lists in various formats.

Many certificate aware clients (like Microsoft Outlook and Netscape Navigator) make use of certificate revocation lists to ensure that certificates are still valid and have not been revoked.

Three links are provided each containing the current CRL in a different format depending on the client. Normal client users would download the CRL in DER format for inclusion in their browser. Web server administrators would use the PEM format. The text format is downloaded as a human readable file.

1.2. User

This section allows the user to manage their certificates. It allows certificate request, retrieval, testing and revocation.

1.2.1. Request a certificate

This link presents the user with a page offering a number of certificate request methods. There are subtile differences between methods which are described below. Each one of the links will take the user to a form. The user will fill in the form and submit the data. The data submitted will be used to create a certificate signing request (CSR) which will go to the certificate Authority to sign and return as a certificate.

The form data has the following fields:

- E-Mail: The email address associated with the certificate

- Name: The name of the user
- Certificate Request Group: This is usually the department or sub group the user belongs to
- Role: The certificate role within the hierarchy, this is usually "User" for most normal users
- Registration Authority: This is usually the physical location at which the user is to be identified (e.g. Personnel)
- PIN: A password used to verify the CSR
- Key Size: The size of the key used in the CSR (Usually set to 1024

After submitting the form the next set of screens the user sees will depend on the client being used and the type of request selected. After the CSR has been generated and submitted the user will be issued with a Certificate Request ID. This takes the form of an integer number. It is important that the user notes this number down as it is required when retrieving their certificate.

Once the user has requested their certificate the Certificate Authority will process the certificate request. This may involve a face to face identification of the user at the Trust Center. When the certificate has been created the user will be informed by email. This email will also include a Certificate Revocation Number (CRIN), this number should be kept in a safe place as it will be required if the user to needs to revoke their own certificate in the future.

1.2.1.1. Request a certificate with automatic browser detection

By pressing this link, OpenCA will try to determine what browser the user is using to request their certificate. Once this has been established the CSR form is presented to the user. The CSR (along with the associated private and public keys) will be generated by the user's browser and submitted to the Certificate Authority.

1.2.1.2. Basic Request

This link leads to a server side key and CSR generation. A user would use this link if their browser did not support CSR generation, of if for some reason they wanted the Certificate Authority to generate the keys and CSR (e.g. For key backup on the server).

1.2.1.3. Netscape's Request

This link should be used if the client is a Netscape type browser (e.g. Navigator or Mozzilla). The CSR generated by the client will be of the type SPKAC.

1.2.1.4. IE Request

This link should be used if the client is an Internet Explorer type browser. The CSR will be generated by the client.

1.2.1.5. Server Request

This link is used to submit a web server certificate request. This is slightly different from a normal client certificate request as the CSR will have already been generated at the web server. There is a field used to upload the CSR, so the user must make sure that they have a CSR to upload before selecting this option.

1.2.1.6. Token Request

This link is the same as the "Basic Request" in that the keys and CSR are not generated at the client. This request is used when the Certificate Authority is going to create the key pair and certificate on a hardware token. You only enter your data and the data is stored at the server but no cryptographic operations take place without operator interaction. In simple terms it is like an email with "Hello, I need a cert. Sincerly your Jon Doe".

1.2.2. Get Requested Certificate

This link provides the mechanism for a user to retrieve the requested certificate and install it in the browser.

The user is presented with a screen and a set of instructions. The most important being that the user must be using the same computer that was used to request the certificate. This is important because both IE and Netscape type browsers need to link the certificate back to the CSR and private keys, this can only be done if the computer that was used to generate the CSR is used to retrieve the certificate.

The user should enter their "Serial Number" in the space provided. The serial number can be:

Certificate's Serial	The serial number of the certificate signed by the Certificate Authority
Request's Serial	The serial number of the submitted request issued to the user at CSR submission time
Your ID:	This is the ID which i used for the batchprocessor. Usually this ID is an account but the ID was defined by the administrator of the batchprocessors.

Upon pressing the "Continue" button, OpenCA attempts to install the certificate into the user's browser. The screens presented to the user depend on the browser being used.

1.2.3. Test Certificate

By pressing this link the user is presented with a screen displaying the session server and client certificate details. In most cases this screen will only display the details of the web server certificate used to secure the session (as this screen is not usually access via pages requiring client side authentication).

The user is offered the opportunity to "Sign" a set of data to test the client certificate. Upon pressing the "Sign" button the user is asked to choose the certificate they wish to use to sign the test data. Once they have chosen their certificate they may be asked to enter the pass phrase securing their private keys (this depends on how the user installed the certificate and private keys during key generation time). Once they that completed this the results of the signing process are displayed.

1.2.4. Revoke Certificate

This screen gives the user the opportunity to revoke their own certificate. To do this they need to fill in the form and press "Continue". The Certificate Serial number can be obtained by examining the certificate (using browser functions) or by looking up the certificate in the valid certificates list. The CRIN code was sent to the user at certificate creation time.

1.3. Certificates

This set of options provides the user with lists of certificates in various states, valid, expired, revoked and suspended. It also provides an interface for the user to search for a certificate.

1.3.1. Valid

Following this link the suer is presented with a screen displaying all valid certificates. The screen shows 20 certificates at a time. The user can scroll through the valid certificates by using the "Extra References" link in the top right of the screen.

For each certificate the screen shows:

Serial	The serial number of the certificate

Common Name	The common name associated with the certificate
Issued On	The date and time the certificate was issued
Email:	The email address in the certificate (the user can click this link to mail the certificate holder)
Role	The type of certificate (e.g. User)

A user can view the content of a certificate by clicking the serial number of the certificate they wish to view. OpenCA presents a screen displaying the certificate details. At the bottom of this screen are two new links where the user can download the certificate (and install it into their browser) or initiate the revocation procedure (in order to do this the user must have the CRIN number for the certificate being viewed, this number is presented to the certificate holder at certificate creation time, so only the certificate holder can revoke their own certificate).

1.3.2. Expired

Clicking this link shows the user a list of all the expired certificates. The screen shows 20 certificates at a time. The user can scroll through the expired certificates by using the "Extra References" link in the top right of the screen.

1.3.3. Revoked

Clicking this link shows the user a list of all the revoked certificates. The screen shows 20 certificates at a time. The user can scroll through the revoked certificates by using the "Extra References" link in the top right of the screen.

1.3.4. Suspended

Clicking this link shows the user a list of all the suspended certificates. The screen shows 20 certificates at a time. The user can scroll through the suspended certificates by using the "Extra References" link in the top right of the screen. Suspended certificates are certificates that have had the revocation process started but not yet revoked by the Certificate Authority.

1.3.5. Search

This link provides a screen to enable users to search for a certificate on the system. The screen allows the user to search based on the criteria of name, email or distinguished name. Wild cards are allowed (e.g. Chris*) in each of the fields. You do not have to fill in each of the fields for the search function to find a match, but the more search data you enter the finer the granularity of the search.

1.4. Requests

This section displays outstanding requests. New certificate requests and revocation requests can be displayed.

1.4.1. Certificate Requests List

Following this link the user is presented with a list of all the current certificate requests at the Registration Authority. The screen shows 20 requests at a time. The user can scroll through the list by using the "Extra References" link in the top right of the screen.

1.4.2. Certificate Revocation Requests List

Following this link the user is presented with a list of all the current certificate revocation requests at the Registration Authority. The screen shows 20 revocation requests at a time. The user can scroll through the list by using the "Extra References" link in the top right of the screen.

2. Registration Authority

This section describes the registration authority interface to the OpenCA PKI. From these screens an RA Administrator can manage certificate requests, view certificate information and manage the RA server.

Each one of the headings below coresponds to a section in the left hand menu pannel of the default RA screens.

2.1. Administration

2.1.1. Server Management

Pressing this link takes the user to the RA Node interface. From here the RA user can control data flow to and from the RA.

2.1.2. LDAP Admin

Pressing this link takes the user to the LDAP Administration interface. From here the RA user can control the import and deletion of data from the LDAP Directory (if it is configured).

2.2. Pending Requests

2.2.1. Certificate Requests

This link displays the user submitted requests and enables the RA Administrator to presses them.

Pressing this link displays a main screen giving the user a drop down list of Registration Authorities to act as. Depending on the configuration "Trustcenter itself" is the default. Clicking the "Continue..." button displays a list of the certificate requests at the selected RA.

Each one of the requests must be processed in turn. By clicking the serial number of the request the user is presented with the details of the request. Four options are then available to the RA User:

2.2.1.1. Edit Request

Pressing this button allows the RA User to edit the details of the request. The editable fields are; Subject alternative name (this is usually defaulted to the supplied email address, but can contain other fields), Subject (or the DN) and Role (or certificate type).

2.2.1.2. Approve and Sign Request

Pressing this button allows the RA User to approve the request and use a certificate to sign this approval. Upon pressing the button the RA User is presented with a list of certificates with which to sign the request approval. Note, if the requests are going to be processed on the CA as a batch process, then each request must be signed with a valid RA certificate (signed by the certificate authority).

2.2.1.3. Approve Request without Signing

Pressing this button approves the request. Note, this can potentially be dangerous as the CA Administrator will have to make a trust decision to process the request or not. If the approved request was signed by a valid RA cert then this decision is unnecesary.

2.2.1.4. Delete Request

Pressing this button deletes the request from the system. Be careful when using this option as there is no warning, the entry is just deleted.

2.2.2. Revocation Requests

2.3. Information

This section allows the user to look at certificates and requests in all states.

2.3.1. Requests

2.3.2. CRRs

2.3.3. Certificates

2.3.4. CRLs

2.4. Utilites

2.4.1. Search Certificate

3. Registration Authority Node

The RA Node is the interface used to control RA operations that deal with external interfaces, for example exporting request data to the Certificate Authority.

3.1. Gateways

This section lists the OpenCA components.

3.1.1. Certificate Authority

This link takes the user to the main CA Server page. This link is only available if the CA is accessable. In the normal OpenCA configuration the CA is offline and so this link will fail.

3.1.2. Registration Authority

This link takes the user to the top Registration Authority page.

3.1.3. LDAP

This section takes the user to a screen to manage the LDAP server, if one has been configured. The following options are available.

3.1.4. Public

Pressing this link takes the user to the OpenCA public interface. This interface is described elsewhere in this document.

3.2. Administration

This section lists the options available to configure and maintain the RA Node.

3.2.1. Server Init

This screen is used to set up your OpenCA RA. It is intended that the screen be used once and once only. There are two links:

3.2.1.1. Initialise DataBase

The RA Administrator should press this link to run the data base initialisation script. Note if you run this script on an existing data base then you are likely to loose all existing data. Be careful !

3.2.1.2. Import Configuration

This link runs the import process. It requires that a CA export file exists at the appropriate device (or directory). The script will open this file and import the configuration data to the RA.

3.2.2. Dataexchange

This link is used to exchange data with other areas of the PKI infrastructure (e.g. CA). Depending on your implementation of OpenCA, only some of the following sections will apply.

3.2.2.1. Enroll data to a lower level of the hierarchy

It is unlikely that there will be a lower level of the hierachy at the RA.

3.2.2.2. Receive data from a lower level of the hierarchy

It is unlikely that there will be a lower level of the hierachy at the RA.

3.2.2.3. Download data from a higher level of the hierarchy

This section is used to download data from the CA to the RA. In order to use this section, data must have already been exported from the CA. This data is usually stored on a floppy disk. Upon clicking any of the following links the user is prompted "You need to provide a support to proceed (depends on your configuration). Are you sure you want to continue ?". This means that you need to have read access to the device that the exported data is on (e.g. the floppy drive).

3.2.2.3.1. All

Pressing this link imports all the data that has been exported from the CA into the RA.

3.2.2.3.2. Certs

Pressing this link imports only the certificate data from the CA into the RA.

3.2.2.3.3. CRLs

Pressing this link imports only the CRL data from the CA into the RA.

3.2.2.3.4. Configuration

Pressing this link imports only the configuration data from the CA to the RA. This is data like user roles (certificate types).

3.2.2.3.5. Batchprocessors

Pressing this link imports only the batch processor data from the CA. (**** I don't understand why this is necessary).

3.2.2.4. Upload data to a higher level of the hierarchy

This section enables the RA Administrator to export data to the export device ready for import to the CA. Upon clicking any of the following links the user is prompted "You need to provide a support to proceed (depends on your configuration). Are you sure you want to continue ?". This means that you need to have write access to the device that the exported data going to be written to (e.g. the floppy drive).

Note, the export of data is in the form of a delta, i.e. only new or modified data is exported. It is an administration task to modify this behaviour.

3.2.2.4.1. All

Pressing this link exports all new or modified RA data to the export device.

3.2.2.4.2. Requests

Pressing this link exports all new or modified requests to the export device.

3.2.2.4.3. CRRs

Pressing this link exports all new or modified revocation requests to the export device.

3.2.3. Backup and Recovery

This section allows the RA Administrator to backup and recover the OpenCA RA database. It is good practice to perform a data base backup regularly.

3.2.3.1. Backup Database

Pressing this link backs up the database to the export device. Upon clicking the link, the user is prompted "You need to provide a support to proceed (depends on your configuration). Are you sure you want to continue ?". This means that you need to have write access to the device that the exported data going to be written to (e.g. the floppy drive).

3.2.3.2. Recovery Initialize Database

Pressing this link configures the data base for import of data. If you are rebuilding the RA then it is important to press this link.

3.2.3.3. OpenCA::DB - if you use DBM-files

If the RA data base is based on DBM files (i..e the default option), use this link to recover the backup data.

3.2.3.4. OpenCA::DBI - if you use a SQL-database

If the RA data base is an SQL data base (the preffered option) then use this link to recover the backup data.

3.2.3.5. Rebuild OpenSSL's database and next serial number

In order to issue certificate request numbers to users this link must be pressed so that the OpenCA RA resets it's static configuration data based on the imported data base data.

3.2.4. Database Handling

Use this link to update the searchable attributes in the database after a software update. *** does anyone have a better explanation of this function ?

3.3. Utilites

3.3.1. E-Mail new users

Pressing this link sends the "New User" emails out to new users. These emails tell the users that their certificates are aready for collection and gives them a link to the public interface to collect their certificates.

3.3.2. Send a CRIN-mail

Pressing this link sends new users an encrypted CRIN mail. The CRIN mail contains a pin code that the user must enter when revoking their own certificates. The user should be able to decrypt the message as they would have created the private key during their certificate request process. The message is encrypted using the public key in the certificate request.

3.3.3. Delete Temp Files

This is a link to a housekeeping utility to delete temporary files.

4. LDAP Interface

This set of screens controls the uploading of data to the LDAP Directory (if there is one configured).

4.1. Update LDAP

4.1.1. CA-Certificate

Pressing this link uploads the CA certificate to the LDAP server. The main screen shows this process and indicates the upload status and success or failure.

4.1.2. Certificates

Pressing this link uploads the current valid certificates to the LDAP direcrectory and removes revoked certificates from the directory. The main screen shows the status and indicates success or failure.

4.1.3. CRL

Pressing this link uploads the current certificate revocation list (CRL) to the LDAP directory. The main screen shows status and indicates success or failure.

4.2. View CA-Certificates

4.2.1. Valid

This link displays a main screen showing the CA certificate. Clicking on the serial number of the certificate displays a new page showing the certificate details.

As this page is from the LDAP server, there are 4 LDAP related options:

4.2.1.1. Add to LDAP

Wrties the CA certificate to the LDAP directory.

4.2.1.2. Add to LDAP with modified DN

Allows the user to enter a modified DN for the certificate and then publish it to the LDAP directory with the modified DN.

4.2.1.3. Delete from LDAP

Deletes the CA certificate from the LDAP directory.

4.2.1.4. Delete from LDAP with modified DN

Allows the user to enter the modified DN name before deleting the certificate with the modified DN from the LDAP directory.

4.2.2. Certificates Expired

This link displays a list of expired CA certificates.

4.3. View Certificates

4.3.1. Valid

This link displays a list of the valid certificates. Clicking the serial number of a certificate displays the certificate details.

As this page is from the LDAP server there a 4 LDAP related options:

4.3.1.1. Add to LDAP

Pressing this button uploads the certificate to the LDAP directory.

4.3.1.2. Add to LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then upload the certificate to the LDAP directory with the modified DN.

4.3.1.3. Delete from LDAP

Pressing this button deletes the certificate from the LDAP directory.

4.3.1.4. Delete from LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then delete the certificate from the LDAP directory with the modified DN.

4.3.2. Expired

This link displays a list of the expired certificates. Clicking the serial number of a certificate displays the certificate details.

As this page is from the LDAP server there a 4 LDAP related options:

4.3.2.1. Add to LDAP

Pressing this button uploads the certificate to the LDAP directory.

4.3.2.2. Add to LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then upload the certificate to the LDAP directory with the modified DN.

4.3.2.3. Delete from LDAP

Pressing this button deletes the certificate from the LDAP directory.

4.3.2.4. Delete from LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then delete the certificate from the LDAP directory with the modified DN.

4.3.3. Suspended

This link displays a list of the suspended certificates. Clicking the serial number of a certificate displays the certificate details.

As this page is from the LDAP server there a 4 LDAP related options:

4.3.3.1. Add to LDAP

Pressing this button uploads the certificate to the LDAP directory.

4.3.3.2. Add to LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then upload the certificate to the LDAP directory with the modified DN.

4.3.3.3. Delete from LDAP

Pressing this button deletes the certificate from the LDAP directory.

4.3.3.4. Delete from LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then delete the certificate from the LDAP directory with the modified DN.

4.3.4. Revoked

This link displays a list of the revoked certificates. Clicking the serial number of a certificate displays the certificate details.

As this page is from the LDAP server there a 4 LDAP related options:

4.3.4.1. Add to LDAP

Pressing this button uploads the certificate to the LDAP directory.

4.3.4.2. Add to LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then upload the certificate to the LDAP directory with the modified DN.

4.3.4.3. Delete from LDAP

Pressing this button deletes the certificate from the LDAP directory.

4.3.4.4. Delete from LDAP with modified DN

Pressing this button allows the user to enter a modified DN and then delete the certificate from the LDAP directory with the modified DN.

4.4. View CRLs

4.4.1. CRLs

This link displays a list of the certificate revocation lists (CRLs). Clicking the Ver (***is this a bug, I think you should click the CRL serial number ***) of the CRL displays the CRL details lists the revoked certificates.

Clicking on the serial number of a revoked certificate displays the certificate details screen in the same way as pressing the "View Certificates Valid" link.

As this page is from the LDAP server there are 2 LDAP related options:

4.4.1.1. Add to LDAP

Pressing this button adds the CRL to the LDAP directory.

4.4.1.2. Add to LDAP with modified DN

Pressing this button allows the user to change the issuer details before uploading to the LDAP directory.

Chapter 6. Functionality Descriptions

1. CA Initialization

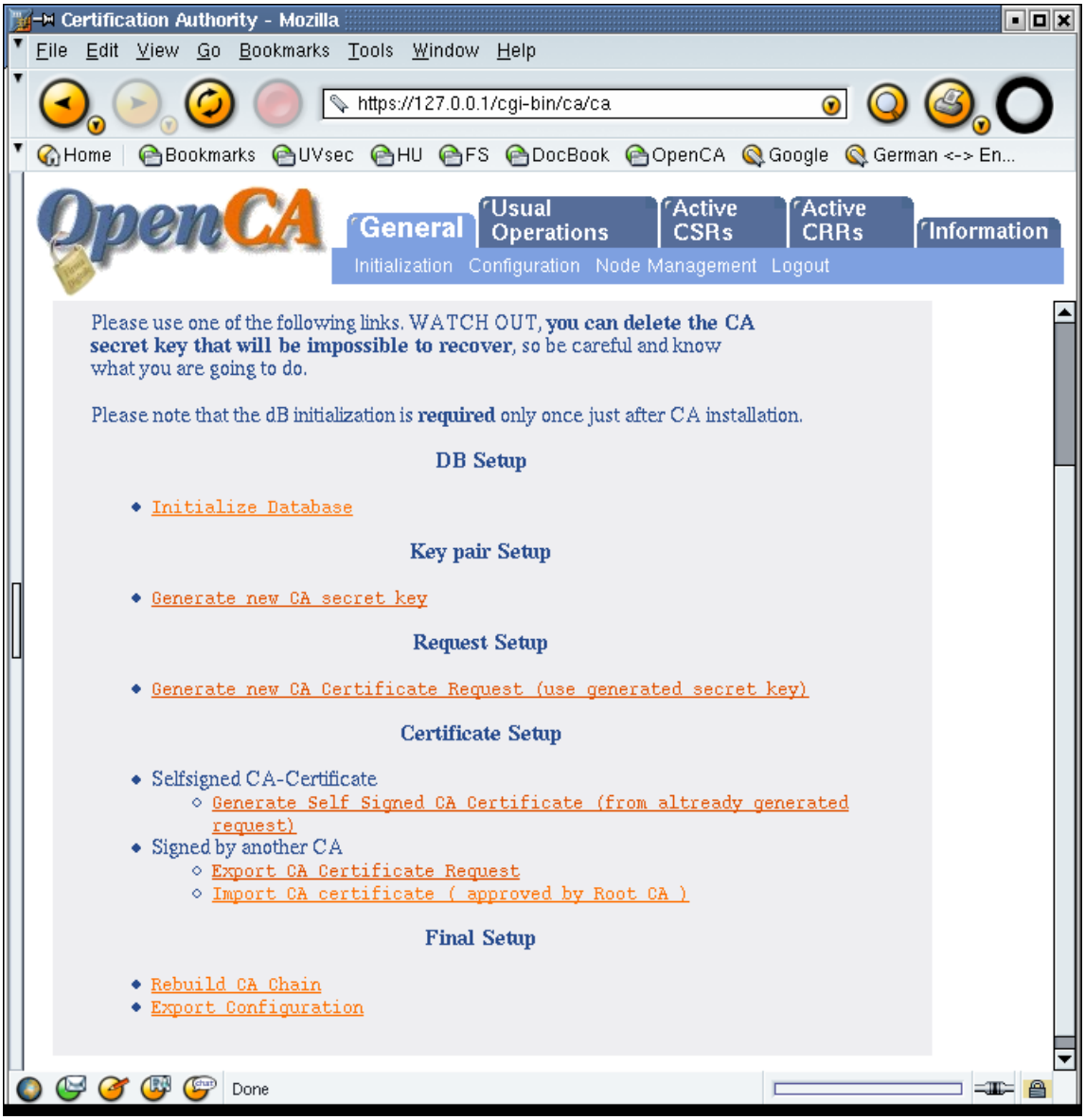
The initialization of the CA consists of three phases and can only be executed once. The first phase is mandatory. It initializes the CA itself. The second and third phase are optional. They create the first two certificates. Phase two creates the first certificate for an operator and phase three creates the certificate for the online web server. First we describe phase one and then we describe phase two and three in one section because they are nearly identical.

Figure 6.1. Phases of the CA initialization

1.1. Phase I: Initialize the Certification Authority

The first phase of the initialization of OpenCA is used to setup all the cryptographic mechanism which are necessary to run a CA. This include things like a private key, a certificate signing request (CSR), a CA certificate and the certificate chain of the CA. If you completed this phase successfully then your CA is ready for operational usage. The other phases only do some optional tasks.

Figure 6.2. Phase I of the CA initialization



1.1.1. Database Setup

This link initializes the database. If you use OpenCA::DB then the backend consists of DBM files. If you use OpenCA::DBI then the backend consists of an SQL database. Today we support MySQL, PostgreSQL, IBM DB2 and Oracle. If you need another database then please contact us.

1.1.2. Key pair setup

Here we do the keygeneration for the CA. You have to enter the algorithm for the key itself, the algorithm for the encryption of the key and the length of the key. If you entered all the cryptographic paramters then you must enter the passphrase if you use an software token from OpenSSL. If you use a hardware token then you must active this token by the appropriate action.

1.1.3. Request setup

If you start creating a new certificate request for the CA certificate then you will be prompted for several informations which will be needed to create a certificate of the style “emailAddress=...,cn=...,ou=...,o=...,c=...”. After you entered all the data OpenCA will display the complete subject of the request. This is the last time when you can modify the subject. If you need another style e.g. dc style then you can enter in this field a subject of your kind. After you entered all parameter for the request the private must be activated (usually via a passphrase).

1.1.4. Certificate setup

There are two general options for a CA certificate in OpenCA. You can use the CA as a root CA then you have to create a selfsigned certificate or you can setup a sub CA then you have to go to another CA and let it sign your request. Both variation need a different handling.

1.1.4.1. Selfsigned CA certificate

If you want to create a new root CA then you have simply to create a new selfsigned CA certificate. This is much more simple then to setup a new sub CA but it is more dangerous. Before you create a new CA certificate please check `OPENCADIR/etc/openssl/openssl.cnf`. The extension are in the section `v3_ca`. It is highly recommend to set the option `subjectAltName` explicitly. If you click on the link then you have only to activate the private key and the rest will be done automatically.

1.1.4.2. Signed by another CA

We talk about a “root CA” in this section but the CA which issues the CA certificate for the new CA which is a sub CA has not to be a root CA. We only use this term to have to different unique namespaces for the CAs in this section.

First you have to export the request to the root CA which has to issue your CA certificate. OpenCA will create a tar file on your export media. This tar file contains a file `careq.pem`. This file is your request in PKCS#10 format. The encoding is PEM (base64). Please go with this request to the root CA and follow its instruction for request processing.

If the root CA issued a certificate for the new sub CA then you have to create a new tar file on your import media. The tar file must contain the file `cacert.pem` which is the new CA certificate. If you click on the link for th import of the new CA certificate then OpenCA copies the file to all necessary places.

1.1.5. Final setup

The last steps can also be done on the interface for the nodemanagement but it is a good idea to do it during the intialization to get a consistent state. The rebuild of the CA chain is necessary to verify digital signatures correctly. If you want to setup a sub CA then you must add all CA certificates of the CA chain in PEM format to the directory `OPENCADIR/var/crypto/chain/` before you rebuild the chain.

The really last step is the export of the configuration to the online server(s). The most OpenCA users ignore this step and handle all the communication between the different nodes of the PKI hierarchy via the interface for the node management. If this is you first OpenCA usage then you should export the configuration and import it into the online server.

1.2. Phase II and III: Create the initial administrator and RA certificate

These two phase are used to create the first certificates of the infrastructure. You can create these certificates via a seperate RA and public interface too but usually the people want only to setup one online RA and this RA should run

with https. So they need at minimum a certificate for the RA's web server. If they use an online RA and it should be a secure solution then the operators must sign a request if they approve a request. So they need an initial user certificate. If the operators have to login via a digital signature and not with login and passphrase then the first user certificate is mandatory. If you compare the two screenshots then you will see that only the names of the forms differ.

Figure 6.3. Phase II of the CA initialization

Figure 6.4. Phase III of the CA initialization

Both phases use the same process like for every normal certificate signing request. Sou can simply read the user guide section where the different requests and their handling will be described. The phases consists of four steps:

Create a new request — This link uses the automatic browser detection of OpenCA so the key and request will be generated by the browser. If you want to use smartcards for the user certificates then you can create the first keypair on a smartcard too if you use the PKCS#11 or CSP drivers of the vendor. It is recommended to use the role “CA operator” for the first user certificate.

Edit the request — Like editing on the RA. See Section 3, “CSR Handling - a request HOWTO” for more details.

Issue the certificate — Like the issue link if you view a certificate on the CA. See the request handling in the user guide.

Handle the certificate — See Section 4, “Certificate Handling” for an explanation of the options.

2. Node Initialization

After you setup an offline CA you usually wants to setup one or more online servers to publish certificates and CRL, receive requests and offers a lot of other services to your customers. To do so you have to install the necessary components plus a so called node interface. This node interface is used to manage the communication in the PKI hierarchy between the different servers and to manage the server itself. If you initialize such an online component then you have to use this node interface.

After you logged in you have to go to and then to . First you should initialize the database and then you have to import the configuration from the CA or any other server which is at a higher position of the PKI's hierarchy. The last step also creates things like a working certificate chain.

3. CSR Handling - a request HOWTO

Usually an administrator or an user suddenly needs more security or is shocked how high does it's risk be if he wants to request a certificate. Then the customers try panically to request a certifcate and don't know how. This is the moment when you should be ready to present them a small explanation how the certification works. We want to give you here an overview how the complete process of OpenCA works. You can simply use this as a base for your user information. The explanations are divided into five plus two parts:

1. how to submit the new request
2. how to edit a request
3. how to approve a request
4. how to issue a certificate
5. how to enroll a certificate
6. how to delete a request
7. explanations of the input fields

Please remember that a request is more correctly a certificate signing request. There is a second request type too - the certificate revocation request (CSR).

3.1. Ways to request a certificate

X.509 certificates are very well known and the format is really established but in the first years there are some problems with the requests because there was no simple protocol to submit requests via clientinterfaces. The result is that there is one standardized format for requests and one proprietary format too. The standard request format is PKCS#10 which is supported by all servers (e.g. web, mail, VPN) and Microsoft Internet Explorer. Netscape developed an own format called SPKAC. This proprietary format is used today by Mozilla and Opera. Surprisingly there is software available which uses certificates and private keys to encrypt data like emails or https-connections but is not able to create a private key or a request. Such a software is KDE's konqueror today. This software simple want to load prepared private key and certificate.

If you think “Wow what a complex area!” then please are not shocked but I don't talked about the request generation until now. The request is only the data but we need something like a procedure for creation and transport of the request too. The browsers know two ways today - Microsofts CAPI (Microsoft Internet Explorer) and Netscape's tag “keygen” (Mozilla, Netscape, Opera). A third way is the way of konqueror which expects we do all for it. A second big area are servers. Some servers like Apache has the same behaviour like konqueror. So you have generate the keys and requests by hand or go to the PKI's public interface and let the PKI does the job. Some other servers like VPN server can generate private keys and requests by itself. Servers and VPN clients support in contrast to webbrowsers two transport protocols - the manual way and SCEP. Usually you can copy the request to a floppy and go to your PKI or you use SCEP to handle all the communication with the PKI automatically.

Now is the moment to calm down and realize that we try to cover all these aspects and you have only to know what you have for a software and what you want to do. The following sections describe the steps for the different kinds of requests and submission technologies. If you have an Internet Explorer or need a certificate for Microsoft Outlook

(express) then you must only read the section for the Microsoft Internet Explorer and so on. If you don't know what to do then simply use the interface with the automatic browserdetection.

3.1.1. Microsoft client request

You have to enter all the informations on the first screen. Please use only keylengths of 512, 1024 and 2048 bits. Other keylengths are actually not supported. It is highly recommended to use at minimum 1024. We recommend that you use 2048 bit because of the last theoretical papers on RSA algorithm. 1024 bits are still safe but it can happen that 1024 will cracked in the next years. If you entered all informations correctly and click ok then you will see a second screen. If you do something wrong then you will see the first screen again.

The second screen shows you the entered informations again. Please check the details. If you click ok then your Internet Explorer will start to generate a new key and request. If the operation succeeds then OpenCA will show you a success page with the serial number of the request. Please remeber this serial before you get in touch with the registration authority.

3.1.2. SPKAC request

SPKAC request are used by Netscape, Mozilla and Opera. All three browser support the special tag keygen which is used to generate the private key and the request.

On the first screen you have to enter all the informations. The keylength can be ignored because you must select it on the second screen again because there is no way to set a default. If you entered all informations correctly and click ok then you will see a second screen. If you do something wrong then you will see the first screen again.

The second screen shows you the entered informations again. Please check the details. Now you have to select a keylength. It is highly recommended to use at minimum 1024. We recommend that you use 2048 bit because of the last theoretical papers on RSA algorithm. 1024 bits are still safe but it can happen that 1024 will cracked in the next years. If you click ok then your browser will start to generate a new key and request. If the operation succeeds then OpenCA will show you a success page with the serial number of the request. Please remeber this serial before you get in touch with the registration authority.

3.1.3. Pregenerated PKCS#10 request handling

There are two methods to submit a pregenerated PKCS#10 - the webinterface or SCEP. SCEP will be covered completely seperate from this section. If you have a PKCS#10 request then you can upload it to the PKI's database. If it is rejected with an errormessage related to the subject (the distinguished name) of the request and you don't know what you are doing wrong then please contact your registration authority. It is possible configure some restrictions to enforce usable requests.

3.1.4. Request a centrally generated smartcard

This is not a real request type. It is more a webformular that send an information to the RA that you want a smartcard. The request does not contain any cryptographic informations. It is simply used to collect some first informations about you.

3.1.5. Serverside key and request generation

This technology uses the same webpages like Microsoft Internet Explorer or Mozilla but there is a big difference. The browsers will generate the private key and request by themselves and only send them to the server but this interface will only send your entered data to the server and the server will create the key and request. You can use this technology for exaple if you need a certificate for a server and you have absolutely no idea from PKIs or you have a browser like konqueror which can use certificates and keys but is not able to generate requests.

On the first screen you have to enter all the informations. If you choose a keylength then it is highly recommended to use at minimum 1024. We recommend that you use 2048 bit because of the last theoretical papers on RSA algorithm. 1024 bits are still safe but it can happen that 1024 will cracked in the next years. If you entered all informations correctly and click ok then you will see a second screen. If you do something wrong then you will see the first screen again.

The second screen shows you the entered informations again. Please check the details. If you click ok then your data will be send to the server again and the server creates a private key and a request for you. Please remember the PIN. It is used to protect the private key. The databases on the server have no backup from this PIN! If all operations succeeds then OpenCA will show you a success page with the serial number of the request. Please remeber this serial before you get in touch with the registration authority.

3.1.6. Automatic browserdetection

The automatic browserdetection is used to determine which mechanism should be used to handle your kind browser. Please use this only if you need a personal user certificate because it really difficult to extract a certificate for a server from a browser and convert it to an appropriate format. If you use an Internet Explorer then please read section for Microsoft client requests. If you use Mozilla, Netscape, Opera or a Mozilla derivate like Phoenix or Firebird then please read the section for SPKAC requests. All other user browser will covered by the section for serverside key and request generation.

3.1.7. Input field explanations

There are several fields which you have to fill in. Here is an overview how they will used:

Certificate data	The input fields in this area are used to generate the subject of the certificate. The subject of a certificate is the name of the certificate.
Additional user data	These informations are only used internally. The entered data is stored in the database of the PKI but they are not part of the certificate. These fields are usually used for things like telephonenumber, addresses and personal IDs.
Role	This is your role in the PKI. A role has several effects. First it decides which extensions will be added to your certificate. The extensions define for applications the certificate can be used. Second it can be used to manage the access rights in the PKI.
Registration Authority	Deprecated. Will be replaced by LOA.
PIN	The PIN will be always hashed and stored in the database. It is planned to use the PIN to identify the issuer of the request because only he can know the PIN. This is not implemented today but the PKI can be configured to use this PIN to allow you start a revocation if you lost the private key and/or certificate. The PIN is called CRIN in this case. If you use serverside key and request generation then this is the PIN for the private key. Only you know this PIN!
Keylength	This is the used length of the private key. Please check this value on the second form too because some browser like Mozilla lost this information if OpenCA asks for your approval for the key and request generation.

3.2. Edit a certificate signing requests

If a user submits a request then this request must be verified by an operator. The operator can look at the request by using the request lists or search the database. If he wants to edit the request then he has to click on edit.

It is possible to change four things in the edit form.

Subject alternative name	The subject alternative name contains additional informations for other softwares which use the certificate but have not the private key. Therefore this alternative name contain informations like emailaddresses, DNS names, IP addresses and more. Emailaddresses are important if you use this certificate for emailencryption because the most clients support only S/MIME v2 and this requires an emailaddress in the certificate. DNS names and IP addresses are useful if you use the certificate for SSL servers or VPNs. If you use a patched OpenSSL 0.9.7 or an OpenSSL 0.9.8+ to issue certificates for Microsoft Smartcardlogin then you have to edit the configuration of the RA. You must add the option for othername to `CSR_SUPPORTED_SUBJECT_ALT_NAMES` and you have to set the correct number of available options at `CSR_DEFAULT_SUBJECT_ALT_NAME_FIELDS`.
Subject or distinguished name	This is the name of the certificate. The least significant information is at top. It is the same order like in RFC 2253 (LDAP string representation).
Role	This is the requested role. The RA operator must set the correct one.
Additional data	These are informations which will assembled only for internal usage. Here you can store such things like contact addresses and special phonenumbers.

3.3. Approve certificate signing requests

If a RA operator is sure that all informations are now correct then he can approve the request. You can do this with and without a digital signature. The digital signature secures the request against manipulation after the approval.

3.4. Issue a certificate from a certificate signing request

If you are on the CA interface and you look at the request then you can only issue a certificate from the request or delete the request. If you click on issue then you have to enter the passphrases or whatever your OpenCA needs to unlock the private key. That's all.

3.5. Certificate enrollment

Please read the section which describes the certificate handling to understand how certificates will be enrolled to the user. If you use a SCEP device then please read the SCEP section.

3.6. Delete certificate signing requests

If you look at a certificate signing request on the RA or CA then you have the option to delete the request. This is necessary if a request has to be rejected by a PKI. If you click on delete then you will be prompted for an additional ok. If you want to make a deleted request valid again then go to the deleted request and renew it.

4. Certificate Handling

This section describes all the things which you can do with a displayed certificate.

4.1. Find a certificate

You can find a certificate with two methods. The first method is search. Go to Utilities --> Search certificate. You can enter some parameters in the displayed search form. The form only accepts wild cards if you use a SQL database. If the search succeeds then you can choose the certificate which will be displayed.

The second method is a little bit more “stupid”. Go to Certificates --> Valid
and try to find the appropriate certificate in the lists. You can navigate by using the links in line `Extra References`.

4.2. Download

4.2.1. Direct Download

You can directly download a certificate into your browser by entering an appropriate serial number. You must know the serial number of the certificate, of the request or you ID in the batch processors. The browser will be automatically detected by the software. Please remember that this method only works if you generated the private key with the browser and the private key is still in your keystore on computer.

4.2.2. Downloads from certificate page

There are three different ways to download a certificate. You can download passive data, or you can download the private key and the certificate or you can install the certificate of another user. If you already have the private key and you want to install a new certificate in your browser then please use the direct download, because this is the only software part which sends special HTML-pages for direct certificate installation.

4.2.2.1. Normal Downloads

If you only need a certificate in a special format then can choose the format and click on Download. The certificate will be send with an appropriate MIME type which prevents browsers from installation. You can save the certificate on a disk and you can do what you (or the policy) want to do with it.

4.2.2.2. Private Key Downloads

If you want to download a certificate and the private key there are two possibilities. If the operation is allowed on your interface and the configuration switch *`REQUIRE_PASSWD_PUBLIC`* is set to `NO` then you can click on download. If you need the key in a format different from PKCS#8 then you must enter the passphrase to convert the private key. After this you will receive the key and certificate and you can save them.

If the operation is allowed on your interface and the configuration switch *`REQUIRE_PASSWD_PUBLIC`* is set to `YES` then you must go to your RA Operator and ask them to set a passphrase. We do this to avoid denial of service attacks against the private key of a user. It is strongly recommended do delete the passphrase after a short period of time and to generate the passphrases with things like **openssl rand**. User or admin “generated” passphrase are often not really secure. If the admin sets the passphrase for this certificate via the RA interface then you can go again to your interface and download the certificate and private key. You have to enter the passphrase for the private key first and then the software will ask you for a second passphrase to grant you access to the export command. If you downloaded the key then please inform the RA Operator and ask him to remove the passphrase to avoid denial of service attacks against you private key.

4.2.2.3. Certificate Installation

Sometimes you need a certificate of another user who has never sent you a signed mail. If you have a normal installation with LDAP support then you can search the certificate in the directory. There are installations where this service is not available. In this case you can go to the certificate page and if the appropriate functionality (`INSTALL_CERT`) is ac-

tivated in the configuration then you can click on install, and the certificate will be automatically installed in your certificate store. After this you can use it to encrypt emails.

4.3. Start revocation

4.4. Write an email to the owner

4.5. Informational messages and their meaning

Anybody has too much time?

5. SCEP

The OpenCA team introduced with version 0.9.2 support for SCEP. SCEP was developed by Cisco. The protocol is usually used by VPN products like routers, switches and software clients to submit requests, download certificates and CRLs. We will document here the configuration for known systems to work with OpenCA. The configuration of OpenCA's SCEP service is described in the administration guide Section 8, “SCEP”.

5.1. SSCEP

SSCEP means Simple SCEP client for Unix. It can be used to request certificates if a device doesn't support SCEP but you want to use SCEP. Usually SSCEP is used to test SCEP daemons to work properly.

Our example uses the following configuration (`sscep.conf`):

Example 6.1. SSCEP configuration

```
# URL of the SCEP server.
URL             http://scep.pki.openca.org/cgi-bin/scep/scep

# This is one is needed with all operations.
CACertFile      ./ca.crt-0

# Possible values: yes or no.
Verbose         yes
Debug           yes

# Display fingerprint algorithm (md5/sha1)
FingerPrint     md5

# Private key created with mkrequest
PrivateKeyFile  ./local.key

# Where to write successfully enrolled certificate
LocalCertFile   ./local.crt

# Certificate request file created with mkrequest
CertReqFile     ./local.csr

# Poll periodically for pending certificate (seconds)
PollInterval    60

# Maximum polling time
MaxPollTime     28800

# Maximum polling count
MaxPollCount    256

# Certificate serial number (decimal)
GetCertSerial   1

# Write certificate as
GetCertFile     ./cert.crt

# Write CRL as
GetCrlFile      ./crl.crl
```

First you have to download the CA and SCEP certificate from the SCEP server. Please check that the CACertFile is correct. Sometimes the SCEP certificate is ca.crt-0-0. Now you have all what you need to request a certificate. `local.crt` must contain your request and `local.key` should contain your private key. To avoid that your CA admin think that your are an attacker please poll not faster then all ten minutes (600 seconds).

You can run now **sscep enroll -f sscep.conf**. The result is some debugging output and sscep starts polling until it reaches it's maximum poll time, receives an errormessage from the SCEP server or downloads the certificate successfully. That's all. No magic but a really simple and efficient client interface.

If you don't want polling e.g. if you use SCEP for web server requests then you can start the enrollment and kill the process when SSCEP starts polling. If you know that the certificate is available then you simply start the enrollment

again. SSCEP tries to send the request again but OpenCA detects that the request is already present in the database. So OpenCA reports a successful submission of the request and the first polling of SSCEP ends with the submission of the new certificate. If you think now that this is a good program to implement a batchprocess to generate smartcards etc. then you are right (but OpenCA has batchprocessors too) :)

5.2. NetScreen ScreenOS [http://www.netscreen.com/]

OpenCA's SCEP service is tested with NetScreen NS-208. NetScreen's SCEP implementation sends SCEP messages in base64 but without any newlines. Now OpenCA can handle this too.

First you have to install the complete CA chain. You have to go to `objects` and then to `certificates`. Here you must set the option `Show` to *`CA`*. Now you can upload the CA certificates via `browse` and `load`.

After you uploaded the complete chain please go to the end user CA and click on *`Server Settings`*. The interface is a little bit mistakable because it display the issuer and in this field you find the link to configure the CA. `RA CGI` and `CA CGI` must be set to OpenCA's SCEP interface. The address is something like *`http://scep.mypki.org/cgi-bin/scep/scep`*. If you want to be consequent then please check the `advanced settings` to be correct for your environment. It is recommended to set at minimum the field `Certificate Renew` to seven days. Finally click on `OK` to save the settings. This can take some time.

Now it's time to make the request. Change `Show` from *`CA`* to *`Local`* and click on `New`. Enter all required informations and choose at minimum a keylength of 1024 bit - smaller keylengths are a security risk. If you finished then click `Generate` to create the key and request. Due to the slow hardware it can take some time. If you see the request then select the checkboxes *`Automatically enroll to`* and *`Existing CA server settings`*. Select the appropriate CA which you configured for SCEP and click on `OK`. This will submit the request.

If the certificate was issued go to the web interface of your NetScreen box. Go to `objects` and then to `certificates`. Here you must set the option `Show` to *`Local`*. Click on `Retrieve` to check for the certificate.

5.3. F-Secure VPN+ [http://www.f-secure.com/support/technical/vpn+/index.shtml]

The SCEP compliance with OpenCA was tested with F-Secure Management Agent 5.02 (FSMA) and F-Secure VPN+ 5.43 (Gateway and Client).

If you want to use SCEP with VPN+ then you must set the appropriate policy in FSMA. Go to **/Settings/Certificate Handling/Enrollment/Active protocol**. There you can choose which enrollment protocol you need. Simply click on SCEP to activate SCEP.

After the activation of SCEP you have to configure it. The configuration settings can be found under **/Settings/Certificate Handling/Enrollment/Protocol Settings/SCEP**. The options are

CA Server URL	The value should look like *`http://scep.mypki.org/cgi-bin/scep/scep`*. This is the default SCEP gateway of OpenCA.
CA Identifier	This parameter is not used by OpenCA. You can ignore it.
Challenge Password	We don't use the challenge password today. So you can ignore it too.
Alternative Name Encoding	The RA interface of OpenCA allows the editing of all parameters of the request except the public key. So it is not important to select the right option here. Nevertheless it is recommended to use *`Only in SubjectAltName extension`* because this avoid the duplication of data in the subject of the request (distinguished name). VPN+ adds the VPN+ identity via the common name to the subject and this usually duplicates the entries in the DN. The recommended option let VPN+ create a CSR which has the

identity in the subject (DN) and the attributes like emailaddress, IP address and DNS name in the subject alternative name.

That's all of the SCEP specific stuff in VPN+. You can read this description with some more details in F-Secures documentation too. You can find some docs on your CD-ROMs and some old docs online at F-Secure.com [http://www.f-secure.com/support/technical/vpn+/index.shtml].

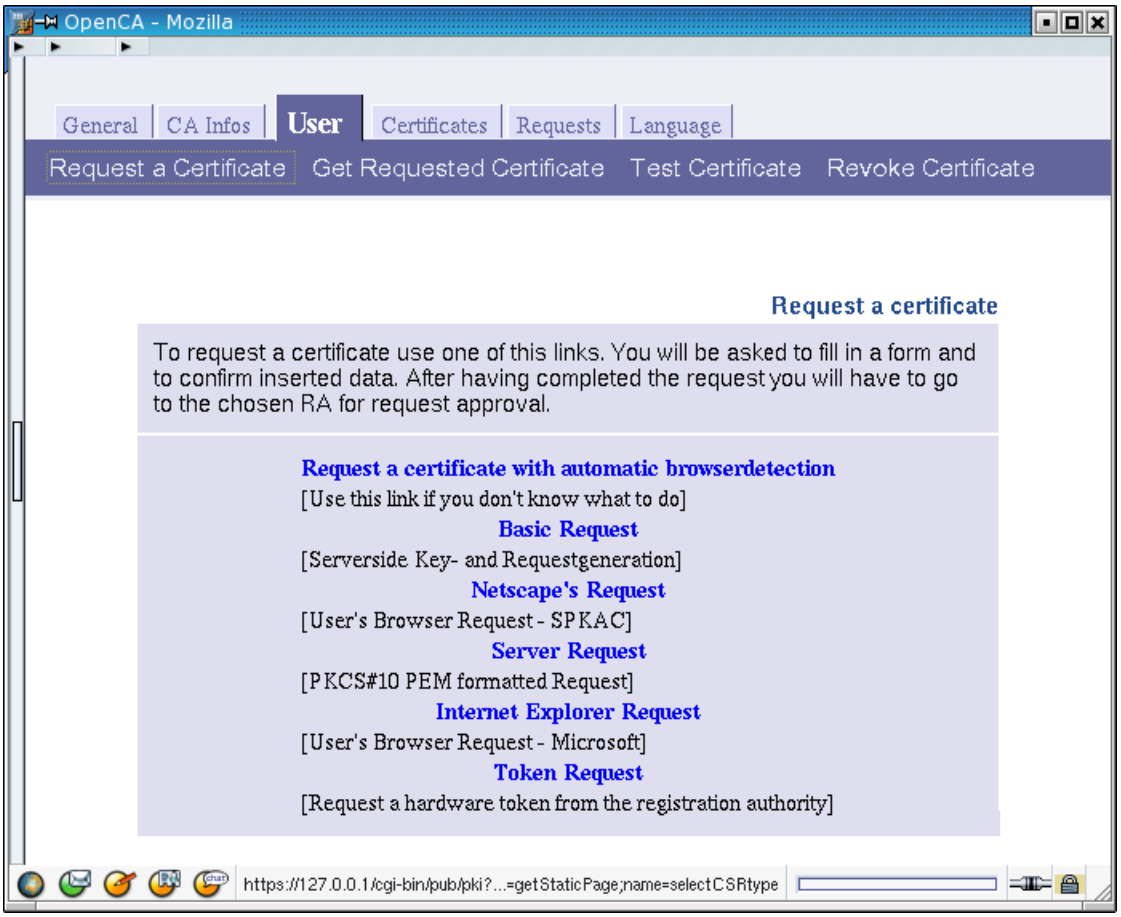
Chapter 7. Client Support

1. Introduction

The biggest problems of PKIs are like for all other big infrastructures the clients. The big advantage of PKIs is that all vendors accept the basic standards. The big disadvantage is that the area of PKI-enabled clients changes very fast and some vendors are in a very early implementation phase or have products with different behaviour.

If this sounds confusing then you can believe us it is confusing. Therefore you can find here some descriptions for the different browsers and the common problems with these browsers. If you have additional problems or other informations then please mail us. We try to support as many client as possible but sometimes a new release can outdate a complete description.

Figure 7.1. Request a certificate



2. Mozilla

This section describes the behaviour of all spin offs from the original Netscape browser. This includes in the security area Opera because Opera includes the crypto stuff in the same way like Netscape. This doesn't mean that Opera includes code from Netscape or Mozilla. Opera only operates in the same way and therefore we included it here.

2.1. General

2.1.1. Requesting a certificate

If you use a Mozilla-like browser and you want to request a certificate from an OpenCA-based PKI then you have to go to the public web interface and klick onto the User area. There you can choose an option `request a certificate`. After this you can choose between several options (see Figure 7.1).

You can choose the link for the automatic browser detection or more error proof the link for SPKAC-based requests. SPKAC is a special format defined by Netscape. This format is used by Netscape, Mozilla and Opera.

Please fill in your data at the next page. If you submit your data and the software finds no mistakes then you will see the data again. Please check them carefully to avoid additional work for your registration authority. If you now submit the form again then your browser generates a new private key and creates a new request with this key. This request will be send by your browser to the web server from your PKI.

Please print the last page or at minimum write down the displayed informations about the necessary procedure and the displayed serials. These serials are necessary to install the later issued certificate.

2.1.2. Installing a certificate

2.2. Mozilla

2.2.1. Backup a certificate

2.2.2. Signing Data

2.2.2.1. SECCLab

2.2.2.2. WaMCom

2.2.3. Mozilla 1.0 and Netscape 6 and 7

2.3. Netscape 4

2.3.1. Backup a certificate

2.3.2. Signing Data

2.4. Opera

3. Microsoft

3.1. Internet Explorer

3.1.1. Requesting a certificate

3.1.2. Installing a certificate

3.1.3. Backup a certificate

3.1.4. Signing Data

3.2. Outlook

3.3. Outlook Express

Part IV. Technology Guide

Table of Contents

Preface xci
8. Introduction 92
    1. Slotechnology 92
9. XML 94
10. Cryptolayer 95
11. Accesscontrol 96
12. Logging 101
13. Webinterfaces 103
    1. Interfacebuilding 103
        1.1. Technology overview 103
        1.2. Customization capabilities 105
    2. CSS 106
    3. Configuration after installation 106
14. Hierarchy 107
    1. Nodemanagement 107
    2. Dataexchange 107
15. LDAP 108
    1. Supported options in interface configurations 108
    2. LDAP schema specification 109
        2.1. Used objectclasses 109
        2.2. Supported attributes 110
        2.3. Common definitions for distinguished names 110
        2.4. Special definitions for user certificates 111
    3. Sourcecodeorganization 112
        3.1. Structure of the code 112
        3.2. The relevant commands 112
        3.3. export-import.lib 113
        3.4. ldap-utils.lib 113
        3.5. OpenCA::LDAP 113
16. Batch System 114
    1. Requirements 114
    2. Design 114
    3. Data Import 115
    4. Change the workflow 117
    5. Default workflow 119
    6. What about the different crypto tokens? 119
    7. Performance 119
        7.1. PIII 850MHz, 256 MB RAM 119
17. Packaging 121
    1. Common Notices 121
        1.1. Required Perl modules 121
    2. RPM-based system 121
        2.1. RedHat/Feodora 121
        2.2. SuSE 121
    3. Debian 123
    4. BSD 123
18. Software Design (legacy from design guide) 124
    1. Database(s) 124
    2. Interface construction 124
    3. openca.cgi 124
    4. libraries 124

5. modules 124
6. commands 124
7. Dataexchange and Node management 124

Preface

This guide has only one goal, it should explain all core technologies of OpenCA. This is not a guide which explains how you setup OpenCA or what you have to do if your OpenCA installation doesn't work but this guide is a good source to understand the software design of OpenCA.

So if you think there is a core part of OpenCA and you don't understand why it is implemented in this way then it is a good idea to read this guide or to write a new section if you think it should be documented.

Chapter 8. Introduction

OpenCA includes a lot of ideas and we think it is impossible to give a 100 percent complete overview about all technologies and ideas. The development model of OpenCA is an evolutional model. The community develops new features if they need them. There is no strict release schedule or big corporation in the background.

This guide also follows the evolutional development model. It contains descriptions of new technologies and core concepts of OpenCA. It is impossible for us to document every detail but we want to give all developers a chance to now what OpenCA is. The developers of stylesheets doesn't develop necessarly the code for the encryption tools but sometimes they want to how they can use encryption. In this case they can take a first look into the Tech Guide to understand the general concepts.

Today this guide includes the following topics:

- Chapter 9, *XML*
- Chapter 10, *Cryptolayer*
- Chapter 11, *Accesscontrol*
- Chapter 12, *Logging*
- Chapter 13, *Webinterfaces*
- Chapter 14, *Hierarchy*
- Chapter 15, *LDAP*
- Chapter 16, *Batch System*

1. Slotechnology

Before we start with concrete technologies it is necessary to describe a general technology - slots. Perl's DBI is a good example howto integrate several database drivers without installing and using more modules then required. This is what we call slot technology.

If you have a module and it needs a module which provides a special interface then it calls a "supermodule" which loads the required module and return it or handle all requests for the user including the management of the loaded modules.

The general mechanism is really easy:

```
my $class = "OpenCA::Token::$name";
eval "require $token_class";
return $self->setError ($@, $@)
    if ($@);
my $token = eval {$token_class->new (@_)};
return $self->setError ($@, $@)
    if ($@);
return $self->setError ($token_class::errno, $token_class::errval)
    if (not $token);
```

I hope you see it is really simple to develop a slot concept :-)

The first step is to load the required class. This has the same effect like a statical "use" statement but "use" doesn't work with dynamic classnames.

The second step is the creation of a new instance. Perl's OO interface is really bad compared with python or others like ruby but it works and we have not the time to rewrite the complete code. After the second eval you have a normal object reference and a new "slot" is established.

Please remember these concept at every time you read from a slot based technology.

Chapter 9. XML

This is no real description of a technology. It is more a statement of the development team. All new configuration files will be in XML.

The idea of all the XML files is the possibility to edit the configuration with texteditors and XML editors. So it is possible to develop a graphical interface for the configuration of OpenCA. The real important statement is the DTD. Today there is DTD but every XML file must contain <openca> as root element. The element below the root element must describe the type of the configuration (e.g. <token_config>). This makes later integrations of different configfiles really easy.

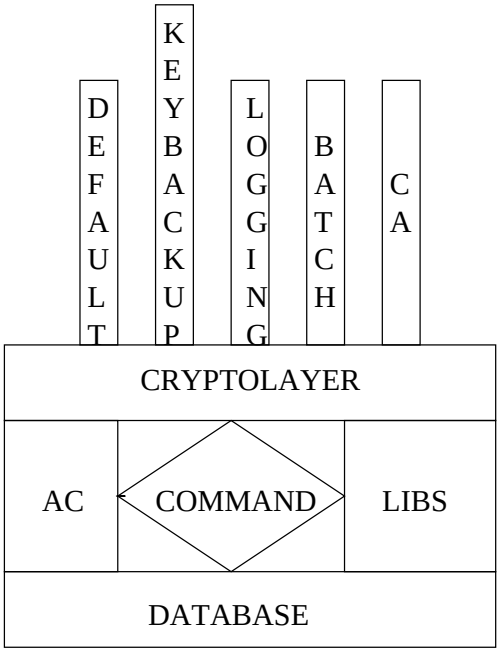
Today we have no DTD but it MUST be written before version 1.0.

Chapter 10. Cryptolayer

The idea for a cryptolayer was born when we started to introduce one new private key every two weeks during the design of the batchprocessor. OpenCA needs today up to four different private keys - CA, logging, batchprocessor, key backup. Additionally we need a cryptoengine for the generation of private keys in the script basic_csr.

We use here the above described Section 1, “Slotechnology” to manage all the different keys. There is a central class OpenCA::Crypto. This class can load and maganage so called "tokens". The software itself uses at every time a special token. Usually there is a default token $cryptoShell avaialable which is not associated to a special key.

Figure 10.1. Example cryptolayer with tokens



If a script or function needs a special key then it calls getToken an receives a loaded object for the appropriate key. The key must be activated by useKey because some tokens can be used without an authentication if the private key is not used. Such a "key" is a software token which is handled by OpenCA::Token::OpenSSL.

The tokens can fallback to the defaulttoken if they don't implemenent a functionality. The token class OpenCA::Token::Empty is only a logical class which automatically falls back to the default token which must be specified in etc/token.xml.

Today there is one additional class beside the classes for empty and software keys - LunaCA3. This class can manage a Luna CA3. This module supports the daemon and session mode too. These modes allow the activation of the module for a complete session or forever (daemon mode). The session ends if the user logs out. This is of course critical but the user decides what he want. The daemon mode activates the HSM and runs it until there is an exlicit shutdown for the HSM via an interface.

Chapter 11. Accesscontrol

During the implementation of some new PKIs several users have serious problems because OpenCA has no own powerful accesscontrol and the implemented accesscontrol uses a proprietary modified Base 64 encoding for filenames. The result of this difficult usage was that the developer itself (me ;-)) don't use this RBAC implementation. So we take the ideas from the first try and start a second XML based try :-)

The accesscontrol has one central configurationfile for every web interface in etc/access_control/. This configurationfile contains references to the token configration and to the access control list if such a list will be used. The file itself has the usual OpenCA structure:

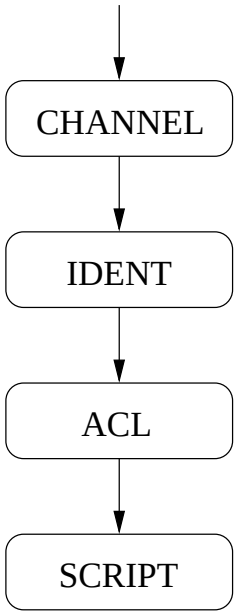
```
<openca>
    <access_control>
        The complete configuration should be here.
    </access_control>
</openca>
```

The problem of the old system was the high complexity therefore it was one main goal to made a modular design that is maintainable by different developers. The system consists therefore of four parts:

1. channel verification
2. login
3. session management
4. ACLs

Every step is a completely isolated pass except of the second and the third step which are unified in the second pass. It is necessary to develop a Section 1, “Slotechnology” which handles the sessions and the data which is stored in the sessions because the actual accesscontrol is only usable with web interfaces but in the far future there should be at minimum a Tk interface.

Figure 11.1. Passes of the accesscontrol



The different passes works like follows.

Figure 11.2. Channel verification

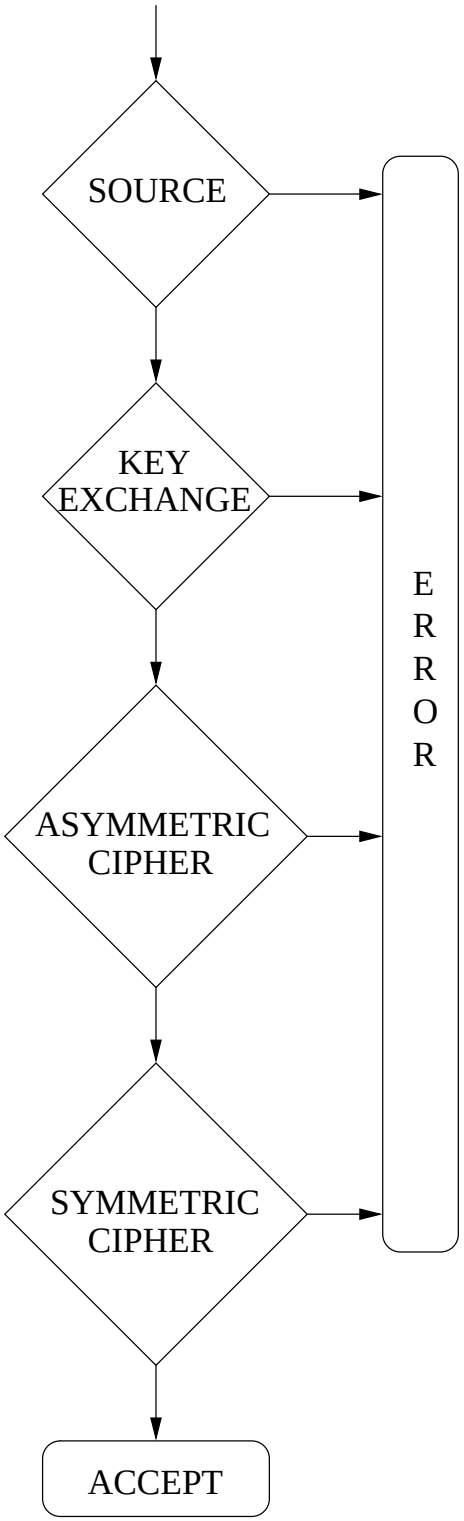


Figure 11.3. Identification of the user

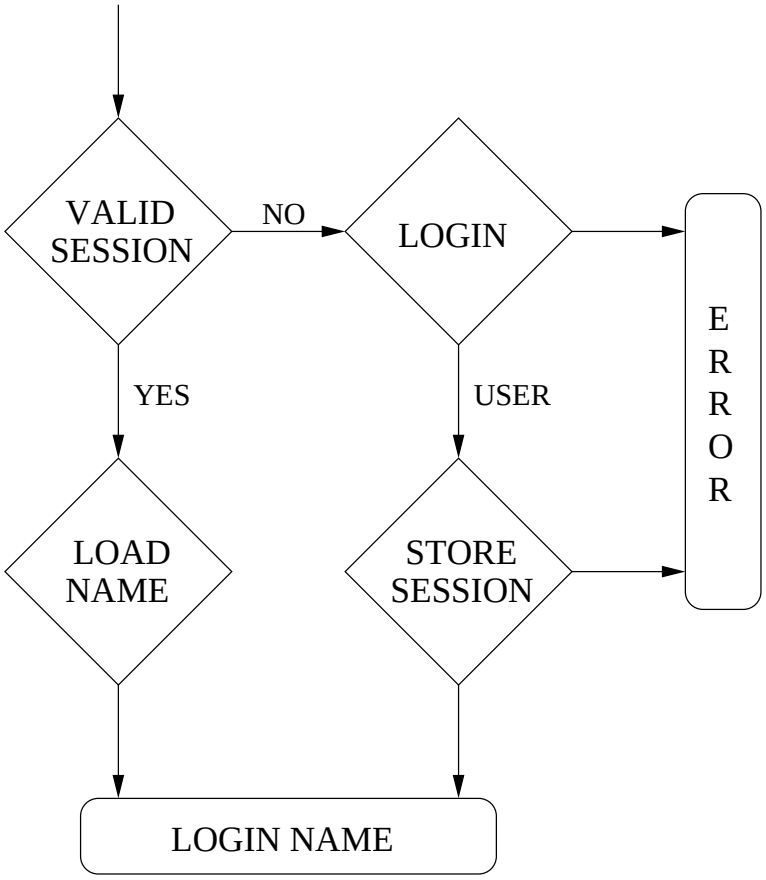
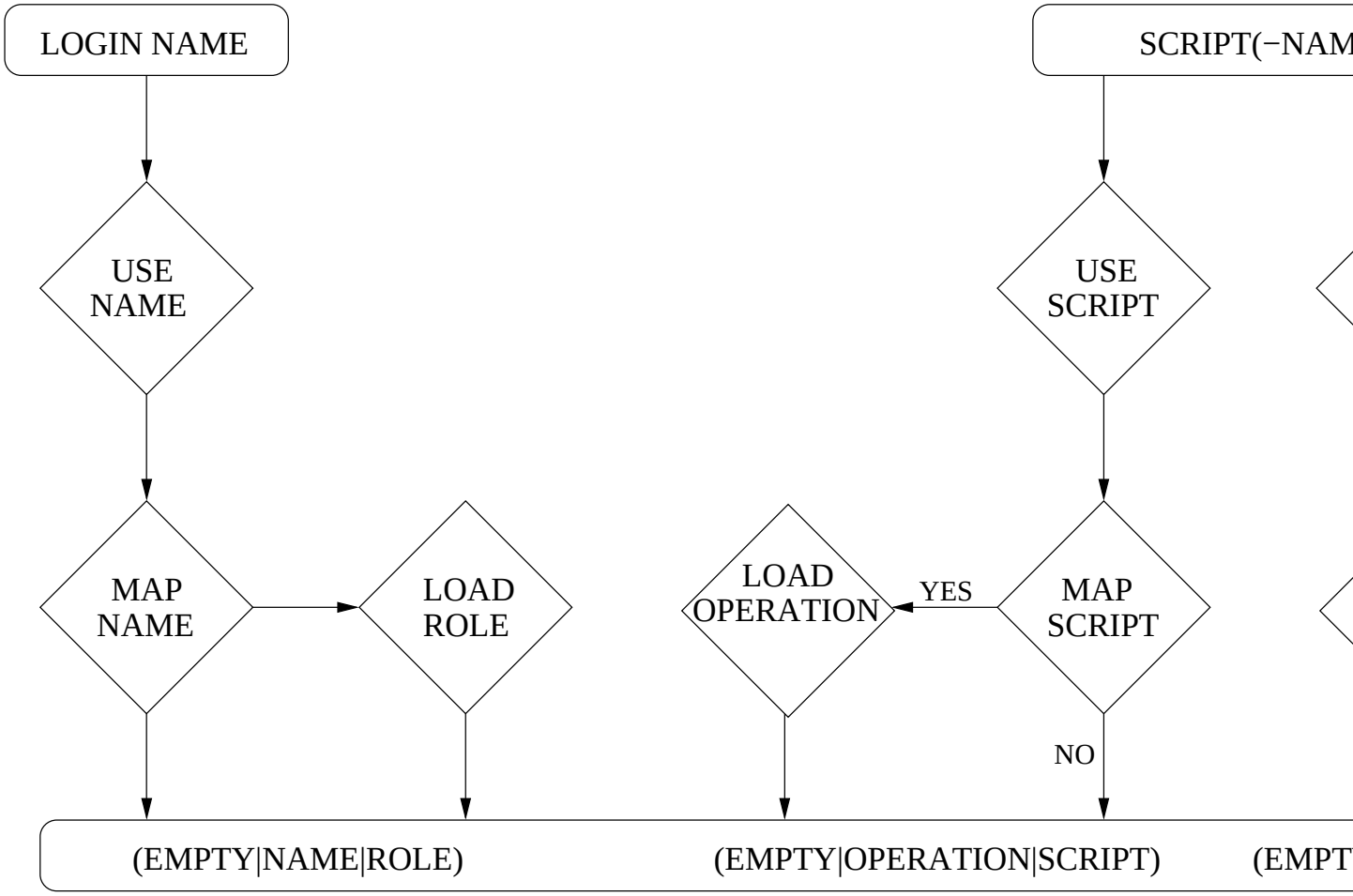


Figure 11.4. Access control list



The exact configuration of the different passes is explained in the administration guide.

Chapter 12. Logging

Logging is a feature which is required for all real security systems. So we see the missing logging as a real problem for OpenCA and so we added it during the 0.9 series. The logging itself based on a slotmechanism to support as many different logging technoligies as possible.

If you want to create a logentry then you have to create a log message first. Such a message is really simple to create:

- Every message wil be build from one hashreference. This reference include values and references to other hashes. So you can set a value via this construct

```
$hash->{user_data}->{emailaddress} = "testuser@abc.com";
```

- The hashreference must include a key CLASS which you can freely choose. This class can beused to filter messages and to decide which slot should be used.
- The hashreference must include a key LEVEL which has to be from one of the following values:
  - EMERG
  - ALERT
  - CRIT
  - ERR
  - WARNING
  - NOTICE
  - INFO
  - DEBUG

  These are the standard Unix levels for messages.
- If you create the message then you have to call new with the parameter HASHREF:

```
OpenCA::Log::Message->new (HASHREF => $hash)
```

If you created a log message then you have to put them into an initialized log module. The module will be intialized by the main scripts like ca and RAServer. The rest will be handled by the OpenCA::Log module.

```
my $log_token = $crypto_layer->getToken ('LOG');
$log_token = $cryptoShell if ( not $log_token );
$log = OpenCA::Log->new (CONFIG => getRequired ('LogConfiguration'),
                         CRYPTO => $log_token);
$log->addMessage (OpenCA::Log::Message->new (HASHREF => $hash));
```

The first line try to load a eventually configured special crypto token for logging. The second line falls back to the default token. This behaviour is used to support HSMs which run in daemon mode and can be used to sign the log. The third line created the logobject and last line add the message.

The configuration of the logging will usually done in etc/log.xml. Today there are two different logger implementation Syslog and XML. You can specify accepted class and level for every configured logger. Class and level accept wildcards.

The Syslog logger needs the type of the used Perl module. You can choose between Sys, Net, and Unix. Today only Sys is really tested. Additionaly you can specify a prefix for every log message. This is useful if you filter your logs with logsurfer or other tools. Please specify a facility too. This allows you to specify the behaviour of the syslogd via syslog.conf. If you use Sys then please specify the used socket type too. I used unix and it works all other types don't work on my linux box but it is better to allow the full flexibility and let it to the user to decide what he needs and uses.

The XML logger only has one option - dir. It specifies which directory the logger should use to establish the needed structure.

Chapter 13. Webinterfaces

1. Interfacebuilding

OpenCA was designed for very fast and radical interface customization. Costumization is understanded as changing the functionality of the interface not the design. The design should be changed via CSS. It was a design goal to allow the user to create an interface for his own special requirements. OpenCA comes with predefined interfaces like ca, ra, public, ldap and node management but the user can merge split or completey mix the functions. This chapter will describe the ideas behind an interface.

There are three mainparts of the customization - statical webpages which show the user all the available functions, the functions itself and links which are only available for some commands.

OpenCA includes also some support for hiding functionalities. The command viewCSR, viewCert and viewCRR can be configured to only show some seletected links. This is not a security issue because it only provides a functionality like linkhiding but it shows the user a consistent interface.

The first section will describe the design to give you chance to understand what is and why it is going on before we describe the possibilities of customization in detail.

1.1. Technology overview

Former versions of OpenCA like 0.9.0 and 0.9.1 use several sheets to support the different commands with templates for the displayed data. This was a nice effort for the first releases but it creates a lot of trouble in some important areas. If you have hundreds of templates then you must translate hundreds of templates. It is really difficult to extract all text informations from templates and you extract really many HTML tags what means that you have style informations in your language database. The next problem is the customization. If you have hundreds of pages and you want to change smething for better CSS stylesheets then you have to change hundreds of templates - really stupid work. So the CCC camp 2003 was the right place to overcome with this solution.

The new solution is the first output module of OpenCA - OpenCA::UI::HTML. It provides OpenCA with four output methods

- libSendReply
- libSendMenu
- libSendStatic
- several logging functions

Let us start with the easiest thing - point four. Nothing changed in this interface except of some classes for the stylesheet. All output pages of OpenCA support stylsheets now. This results in some problems with old Netscape browsers. All actual browsers like Mozilla, Microsoft Internet Explorer, Konqueror etc. have no problems. The old Netscape ones cannot interpret stylesheets correctly. They only output the pure links and not the nice tabbing style like the other browsers but it was time to make a cut and to take up the cudgels for the future (who will use Netscape 4.7 in one or two years?).

`libSendMenu` creates and manages menus from `menu.xml`. You can configure the complete menu in this single file. If you want to hide some complexity then you can modify this XML-file. If you installed OpenCA from scratch then you can search for the pub interface in this file and you will see the raw structure. The DTD is like this:

Gramar of `menu.xml`

```
menu.xml ::=
<openca>
  <interface_menus>
   $interface+
  <interface_menus>
<openca>

interface ::=
   <interface>
     <name>ca<name>
     <htdocs_prefix>/ca<htdocs_prefix>
     <cgi_prefix>/cgi-bin/ca/ca?<cgi_prefix>
     <logo_left><logo_left>
     <logo_right><logo_right>
     <menu>
       $item+
     <menu>

item ::=
       <item>
         <name>General<name>
         ($subitem|$lastitem)
       <item>

subitem ::=
         <id>1<id>
         $item*

lastitem ::=
           <name>Initialization<name>
           <link>cmd=getStaticPage;name=init<link>
           <target>main<target>
```

It is not a real DTD. It is more a grammar. OpenCA support now menus with unlimited depth but you have to take care about the frames of course.

`libSendStatic` is used by **getStaticPage**. Here you can define the functions which the user can see if you show him some statical information. The interface supports NAME, EXPLANATION, TIMESTAMP and ITEM_LIST.

Meanings of `libSendStatic` interface

```
NAME -> simple string
EXPLANATION -> long explanation of this page
TIMESTAMP -> true value, off by default
ITEM_LIST -> double array ($list->[$i]->[$j])
             $j == 0 is th
             $j != 0 is td
```

`libSendReply` is more dynamic than `libSendStatic`. Please check the source code until I checked in the docs tomorrow. It supports command panels and lists, item lists, info lists and siginfo. If you need to build statical pages then please use getStaticPage (see listCSRtype in this file for an example).

1.2. Customization capabilities

1.2.1. Statical Pages

OpenCA still includes some statical webpages which will be created the command `getStaticPage`. If you want to change the content of a static page then please change this file.

1.2.2. cmds

There are two possibilites for additional functions - you can write a new one or you can use and already existing one.

1.2.2.1. New command

If you write a new command then there are two important things. First Every command include a function cmdsFilename. This is the function which is called by the main script if the function filename is requested. Second if you implement a script which has to handle different states then please write a function checkFilename which includes all the status checks to avoid that somebody can start statusinjections.

If you want to establish the new command in the sourcecode then you must add the filename in src/common/lib/cmds/Makefile which installs the file. Additionally you must add the filename to src/web-interfaces/interface/cmds/Makefile. This makefile installs a dynamical link to the file. If this link is not present then the file doesn't exist for this interface.

If you want to add the new command directly in an existing installation then install the file in lib/cmds and add a dynamical link to this file from lib/servers/interface/cmds/.

1.2.2.2. Add an existing command

You can add the command to a preconfigured interface in the sourcecode if you add the filename to src/web-interfaces/interface/cmds/Makefile.

You have to add dynamical link from lib/servers/interface/cmds/ to lib/cmds/ if you want to add the command to an already installed system.

1.2.3. configuration files

There are some scripts which are able to hide references. viewCSR, viewCRR and viewCert are such scripts. There is an option CmdRefs_viewCSR in the config file. There you can specify which functions should be available via this

interface. Please remember that this has nothing to do with security! The access control will be handed the normal access control and by the installation of the scripts. Link hiding is no security. The following options are actually supported:

CmdRefs_viewCSR	EDIT, APPROVE, APPROVE_WITHOUT_SIGNING, ISSUE_CERT, ISSUE_CERT_NEW, ISSUE_CERT_RENEW, ISSUE_CERT_PENDING, ISSUE_CERT_SIGNED, ISSUE_CERT_APPROVED, DELETE, DELETE_NEW, DELETE_RENEW, DELETE_PENDING, DELETE_SIGNED, DELETE_APPROVED, RENEW, RENEW_ARCHIVED, RENEW_DELETED, GENERATE_KEY
CmdRefs_viewCRR	EDIT, APPROVE, APPROVE_WITHOUT_SIGNING, REVOKE_CERT, REVOKE_CERT_NEW, REVOKE_CERT_PENDING, REVOKE_CERT_SIGNED, REVOKE_CERT_APPROVED, DELETE, DELETE_NEW, DELETE_PENDING, DELETE_SIGNED, DELETE_APPROVED
CmdRefs_viewCert	INSTALL_CERT, LDAP, REVOCATION, SENDCERT, SEND_CERT_KEY, VIEW_CSR, TOKENHANDLING, MAIL, SET_PUBLIC_PASSWD, DELETE_PUBLIC_PASSWD

1.2.4. configure_etc.sh

Perhaps a small hint - if you setup more than one public (or any other) interface then please check that **configure_etc.sh** only configures this interface. If you don't check this then **configure_etc.sh** will reconfigure the other public interfacers too and this will crash their paths.

2. CSS

This part should be written by the guy who work on CSS - Massmiliano, Julio, Garreth ...

3. Configuration after installation

The OpenCA project has the goal to develop a trustcenter software which can be used out of the box. We don't ship prepared policies but we want to support the user with simple installation and update routines. This requires that the user can install a RPM or DEB package and then configure the software.

Such a complex software like a trustcenter cannot be configured with one single configuration file but it is impossible to ask the user for the configuration of over 200 files. Therefore we develop a post-install method for a postconfiguration after "configure;make;make install". The result is a single file etc/config.xml and a script etc/configure_etc.sh.

The file contains all basic configuration options which must be changed by the user. You simply configure config.xml and then you run the script which configure all template files.

If a criticism asks now why it is not possible to have one configurationfile if config.xml manages the complete configuration then please remeber the word "basic". Several CA admins have to change the configuration of the extensions for example to support special softwarerequirements. So config.xml is a technology to create useful binary packages but it is necessary to have all the other configuration files to use the full flexibility of the standards.

Chapter 14. Hierarchy

1. Nodemanagement

2. Dataexchange

Chapter 15. LDAP

Be warned - this is a developer documentation which only documents the possibilities and technical background of OpenCA ldap caode but this is not a howto or a user documentation.

1. Supported options in interface configurations

The configuration is splitted into two parts - a OpenCA related part and a LDAP related part. The OpenCA related part conists only of four options - the LDAP activation, automatic LDAP updates during imports and distinguished name manipulations for CA objects like CA certificates and CRLs. These options can be configured in the interface configurations in `OPENCADIR/etc/servers/`.

LDAP — If you set this option to "yes" then the LDAP code will be activated.

updateLDAPautomatic — This option will be used by the node interface. If the value is yes then the LDAP server will be updated automatically during imports of certificates, CRRs or CRLs.

LDAP_CRL_Issuer — Some users want to store the CRL in a special node of the LDAP server which is not identical with the issuer of the CRL. This can be happen if the user specifies a special CRL Distribution Point (CDP) which differs from the subject of the CA certificate. Here you can specify this special distinguished name. Please remember that OpenCA is today not able to add this node automatically if it is not present.

LDAP_CA_DN — Some users want to store the CA certificate in a not standard conform node which means that there is perhaps an already existent directory which conflicts with the PKI structure. Here they can add the distinguished name of this special node. This node can be automatically added by OpenCA.

The LDAP related part of the configuration can be found in `OPENCADIR/etc/ldap.xml`. This central configuration file avoids double configurations which can produce many errors and confusion because you are sure that you changed it but you only did it for one interface. The isolated configration allows better names for the configuration options too.

host — This is the hostname of your LDAP server.

port — This is the port where your LDAP server listens.

suffix/dn — This is the suffix (OpenLDAP terminology) of your LDAP server. You can add here several suffixes if your LDAP server supports this feature (e.g. OpenLDAP v2). Every suffix must be placed in a seperate dn tag. The suffix tag is the bracket for all those suffixes.

Example 15.1. suffix in `ldap.xml`

```
<suffix>
  <dn>o=OpenCA, c=IT<dn>
  <dn>o=OpenCA, c=DE<dn>
<suffix>
```

login	The bind DN of the user which OpenCA uses to add data to the server.
passwd	The passphrase for OpenCA's ldap account.
protocol_version	OpenCA supports LDAP v2 and v3. The default is v2 because all servers can support v2. Several new distributions especially of Linux deactivates the LDAP v2 support. So if your OpenCA LDAP code completely fails check first the protocol versions of OpenCA and your LDAP server. Some other options like ldaptls and ldapsasl require LDAP v3. So be really careful which protocol you use. If your LDAP server supports protocol version 3 then please use it. It avoids a lot of trouble.
tls	Use no or yes to deactivate or activate TLS. Please remember that this option only works with LDAP v3.
sasl	Use no or yes to deactivate or activate SASL. Please remember that this option only works with LDAP v3. We use CRAM-MD5 for passphrase hashing.
excluded_roles/role	OpenCA supports the possibility to exclude roles from certificate publishing. This can be useful for security reason and be required by privacy laws. If you have such a special role simply add it to to this options.

Example 15.2. excluded roles in `ldap.xml`

```
<excluded_roles>
  <role>Internal Security Stuff<role>
  <role>IDS VPN<role>
<excluded_roles>
```

2. LDAP schema specification

2.1. Used objectclasses

Of course we will not list top here.

STRUCTURAL

- country
- device
- inetOrgPerson (inherits from organizationalPerson)
- locality
- person

- organization
- organizationalPerson (inherits from person)
- organizationalRole
- organizationalUnit

AUXILIARY

- dcObject
- pkiCA
- pkiUser
- opencaUniquelyIdentifiedUser
- opencaEmailAddress
- opencaSCEPDevice

2.2. Supported attributes

If you are missing a special attribute for your installation then please contact us (openca-users@lists.sf.net).

- dc
- c
- o
- st
- l
- ou
- unstructuredName
- unstructuredAddress
- cn
- sn
- emailAddress
- serialNumber

2.3. Common definitions for distinguished names

The following table describes what we use for objectclasses and attributes if we insert a node in the LDAP tree. The least significant component of the distinguished name is the "LSC of the DN". This doesn't document the used auxiliary classes.

Table 15.1. Schema usage

LSC of the DN	filled attributes	filled attributes if present	objectclass stack
dc	dc		top, dcObject
c	c		top, country
st	st		top, locality
l	l		top, locality
o	o		top, organization
ou	ou		top, organizationalUnit
unstructuredName	cn	unstructuredName, unstructuredAddress, serialNumber, st, l, ou	top, device, opencaSCEPDevice
unstructuredAddress	cn	unstructuredName, unstructuredAddress, serialNumber, st, l, ou	top, device, opencaSCEPDevice
cn	cn	cn, st, l, ou, mail	top, organizationalRole (opencaEmailAddress)
sn	cn	cn, st, l, ou, mail	top, organizationalRole (opencaEmailAddress)
emailAddress	cn	cn, st, l, ou, mail	top, organizationalRole (opencaEmailAddress)
serialNumber	cn	serialNumber, o, ou, l	top, device

If we add a node to the directory tree then we add at every time to the objectclass stack the classes pkiCA and pkiUser. This is perhaps not the cleanest solution but it is safe for every possible configuration. If we add a node with the class organizationalRole then we add the auxiliary class opencaEmailAddress if an emailaddress is present.

2.4. Special definitions for user certificates

If we add a node for the subject of a normal certificate to the directory tree then we use modified objectclass stacks to support ldap browsers which search for a special emailaddress. We don't do this generally to return only a node of the tree if this node contain a certificate. This was a problem in several old releases of OpenCA.

Table 15.2. Schema usage for user certificates

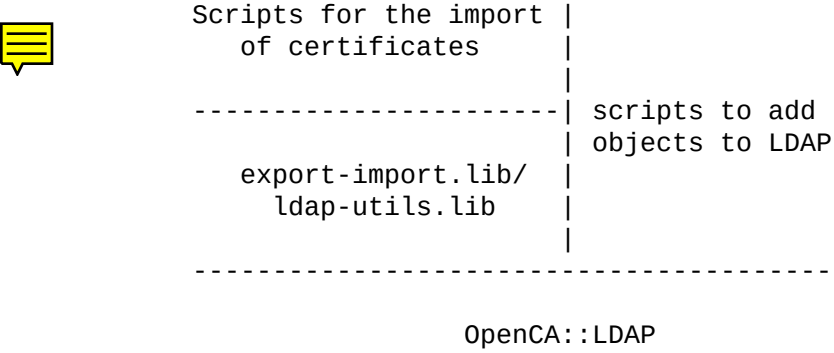
LSC of the DN	filled attributes	filled attributes if present	objectclass stack
cn	cn,sn	mail, o, st, l, ou	top, person, organizationalPerson, inetOrgPerson
sn	cn,sn	mail, o, st, l, ou	top, person, organizationalPerson, inetOrgPerson
emailAddress	cn,sn	mail, o, st, l, ou	top, person, organizationalPerson, inetOrgPerson
serialNumber	cn,sn	mail, o, st, l, ou	top, person, organizationalPerson, inetOrgPerson, opencaUniquelyIdentifiedUser

If the distinguished name doesn't contain an emailaddress but OpenCA detects an emailaddress in the subject alternative name then we use this emailaddress.

3. Sourcecodeorganization

3.1. Structure of the code

Figure 15.1. LDAP source schema



3.2. The relevant commands

- addCertsLDAP (puts all valid certs to LDAP)
- addCrlLDAP (puts all CRLs to LDAP)
- importAllFromCA (via export-import.lib)
- importCRL (via export-import.lib)
- importCerts (via export-import.lib)
- importCertsLDAP (puts all certs from the last import to LDAP)
- importConfig (puts CA-certs to LDAP)
- updateCACertsLDAP (update the CA-certificates on the ldap server)
- updateCRLonLDAP (writes the most actual CRL to LDAP)
- updateCertsLDAP (writes/removes the user-certificates to/from LDAP)
- updateLDAP (puts all certs from the last import to LDAP)

(oh, we have a redundancy here updateLDAP and addCertsLDAP do the same) (updateLDAP is reserved for the future so set all links etc. to importCertsLDAP) (addCertsUser should not be a function of `ldap-utils.lib`)

3.3. export-import.lib

eximObjectToLDAP

3.4. ldap-utils.lib

- LDAP_addCertsUsers (will be moved to importCertsLDAP)
- LDAP_get_crl (determines the newest CRL)
- LDAP_get_ca (determines the newest CA certificate)

3.5. OpenCA::LDAP

- add_object (takes a cert and create the necessary nodes in the LDAP)
- add_attribute (add certs and CRLs to the LDAP)
- delete_attribute (remove certificates from LDAP)

Chapter 16. Batch System

The most organizations which decide to evaluate or use PKIs make a decision We need a PKI. After some tests, discussions and perhaps some failed roll-outs they start looking for automation tools to limit the chances where end-users can make mistakes which are timeconsuming and in this way expensive. OpenCA tries to solve this problem like many other PKI solutions with a flexible batch system.

1. Requirements

The batch system has to fullfill several requirements. It must be

- fast
- easy to configure
- extendable
- adaptable to workflows

These are only some requirements. Usually every organization has a bunch of additional problems which should be solved.

2. Design

The core of OpenCA's batch system is a statemachine which is has primarly nothing to do with OpenCA. This statemachine has a configuration which defines states, functions and some workflows. This statemachine reads at every step the configuration of a user, checks it's actual state, calculates the next function of the workflow which must be executed and returns the function. OpenCA does nothing else then to execute the function for this user. If this sounds really simple to you then please be sure this is only a simplified description :)

First the statemachine doesn't handle users. It handles processes of users. This is necessary because we don't know any installation where every user have only one key and one certificate. At minimum every key must be managed by an own seperate workflow.

Second the statemachine has a two-part configuration. There is a XML file which defines the position of the core configuration files and the options of the batch system functions. The core configuration files are simple text files because they have no structure. They are only lists which are very fast parseable. The core configuration consists of the following parts:

- a list of the existing processes (user + key)
- a list of all known functions
- a list of all known states
- a directory which contain the start configuration of the states for every function

Before you now start testing and playing with the batch system one important advice. If you use the batch system then please notice that the default installation uses the CA token as log token too. This means that an activated CA token sign every log entry. This slows down all operations. So if you think that the batch system is too then please remember the small warning about the log token. The issuing of certificates must be done with the log token of course but the time consuming logging should be acceptable in such a critical moment.

3. Data Import

Before you can use the batch system you have to import some data. There are three different import actions in OpenCA's batch system:

1. new users
2. new processes
3. data for processes

All three formats are described in the following. You must store the three files on your dataexchange media in the files `batch_new_user.txt`, `batch_new_process.txt` and `batch_process_data.txt`. After the specifications you can find an example.

First a description of the formats to create a new user and a new process. Both formats are very simple.

Batch System Import New User Format

```
file   ::= ( serial . "\n" )*
serial ::= this is a user name or ID
```

Batch System Import New Process Format

```
file    ::= ( serial . "\n" . process . "\n" . "\n"+ )*
serial  ::= this is a user name or ID
process ::= this is a process name
```

The format to import the data is a little bit more complicated because it must support some special cases.

Batch System Import New Process Format

```
file    ::= ( ( id . param* . "\n" )*
id      ::= "USER ". serial . "\n" . "PROCESS " . process . "\n"
serial  ::= this is a user name or ID
process ::= this is a process name
param   ::= ( common | loa_mode | key_algorithm |
              key_length | state | subject_alt_name )."\n"
common  ::= name . " " . value ."\n"
name    ::= "SUBJECT" | "ROLE" | "LOA" | __user_defined__
value   ::= value of the parameter

key_length       ::= "KEY_LENGTH" . " " . __unsigned_integer__
loa_mode         ::= "LOA_MODE" . " ". ( "IGNORE" | "NORMAL" )
key_algorithm    ::= "KEY_ALGORITHM" . " " . ( "rsa" | "dsa" )
state            ::= ( "SET_STATE" | "UNSET_STATE" ) . " ".
                     __existing_batch_state__
subject_alt_name ::= "SUBJECT_ALT_NAME_" . san_number . " " .
                     san_name . ("="|":") . san_value
san_number       ::= integer / position in the subject alternative name
san_name         ::= "DNS" | "IP" | "EMAIL" | "OTHERNAME"
san_value        ::= value of the alternative name component
```

Please respect the minimum keylength to avoid trouble. __user_defined__ means that you can store other data in the batchprocessor too. This is required for customized. If you need other data which you cannot import with this system then please write a mail and explain your requirements.

Here you find now a very simple example for the two users Jon and Jane Doe.

Example 16.1. `batch_new_user.txt`

```
jon_doe
jane_doe
```

Example 16.2. `batch_new_process.txt`

```
jon_doe
enc_key

jon_doe
sig_key

jane_doe
key_1
```

Example 16.3. `batch_process_data.txt`

```
USER jon_doe
PROCESS enc_key
set_state new_process
ROLE User
SUBJECT_ALT_NAME_1 email:jon.doe@openca.org
SUBJECT_ALT_NAME_2 email=jon.doe@sf.net
SUBJECT CN=Jon Doe, O=OpenCA, C=IT

USER jon_doe
PROCESS sig_key
set_state new_process
ROLE User
SUBJECT_ALT_NAME_1 email:jon.doe@openca.org
SUBJECT_ALT_NAME_2 email=jon.doe@sf.net
SUBJECT CN=Jon Doe, O=OpenCA, C=IT

USER jane_doe
PROCESS key_1
set_state new_process
ROLE User
SUBJECT_ALT_NAME_3 email:jane.doe@openca.org
SUBJECT CN=Jane Doe, O=OpenCA, C=IT
```

4. Change the workflow

First you have to understand that we implemented something called a finite statemachine. If you don't know what this is then take good books on theoretical computer science or ask some professors. Second you have to understand our implementation :) Ok, you have only to understand the configuration. A finite statemachine consists of states and state transition functions (German: Zustandsuebergangsfunktionen).

All known states are stored in `OPENCADIR/etc/bp/states.txt`. You can change this filename like every other filename too in `OPENCADIR/etc/bp/bp.xml`. All known functions are stored in `OPENCADIR/etc/bp/functions.txt`. The startconfiguration of states which causing a function to start it's actions are stored in the directory `OPENCADIR/etc/bp/functions`. The filenames are `function_name.txt`. Every line contains one state which is required to start the function. Every process of a user has a configurationfile `states.txt` which represents the actual state of the user. It has the same format like the configurationfiles of the functions.

If you want to change the default workflow or create a complete new workflow then you have to do the following:

1. Visualize the state-transition graph.
2. Mark the changes in the graph.
3. Create `OPENCADIR/etc/bp/states.txt` which must include all possible states.
4. Create `OPENCADIR/etc/bp/functions.txt` which includes all available batch functions.
5. Create the start configuration of the states for all batch functions in `OPENCADIR/etc/bp/functions`.
6. Check that the start configuration specified in `OPENCADIR/etc/bp/functions` are conflict free.
7. Please perform the last step twice if you change an already running system to ensure that you don't remove a state which is already present in the system and that you are able to still handle all states.
8. Implement the new and change the old functions.
9. Test the new system before deploying it.

The implementation of a new function is not the simplest task but it is not very difficult. The biggest problem is the impact that a function can create if it is executed in a not expected situation. If it do something wrong then such a function do it usually wrong for every user. The steps to create a new function are the following ones:

1. Add the function to the list in `OPENCADIR/etc/bp/functions.txt`.
2. Create the file `OPENCADIR/etc/bp/functions/function_name.txt` and specify the start configuration of the states there.
3. Create the `OPENCADIR/lib/bp/function_name.sub` by copy and paste from another batch function.
4. Change the function name in the file `function_name.sub` to workflow_function_name.
5. Change the source code of the function to do what you want. The function get two parameters - *`USER`* and *`WORKFLOW`*. You can use these parameters to access the data in the batch system.
6. Finally you have to change the actual state configuration of the workflow.

Please allways log what you are doing. It is the only chance to discover the source of problems. Allways write an entry to the log message if you discover something unusual in a workflow.

5. Default workflow

Table 16.1. Default OpenCA workflow

Function	Start States	Result States on Success	Result States on Failure
create_pin	new_process	new_pin	
check_pin	new_pin	checked_pin	
check_key_params	checked_pin	checked_key_params	
create_key	checked_key_params	new_key	
check_key	new_key	checked_key	
backup_key	checked_key	backuped_key	
check_csr_params	backuped_key	checked_csr_params	
create_csr	checked_csr_params	new_csr	
complete_csr	new_csr	completed_csr	
check_csr	completed_csr	checked_csr	
create_cert	checked_csr	new_cert	

6. What about the different crypto tokens?

First you must have a fundamental understanding about OpenCA's concept of crypto tokens. You can find these fundamentals at . Second the batch system uses up to three different crypto tokens actively and one passively. The three actively used tokens are the CA, the key backup and the batch token (BP token). The CA token is only used to sign certificates. The key backup token is only used to encrypt a keybackup. The batch token is used for all other things like encrypting PINs and private keys. The passive token is the log token which is allways active or never.

All token must be explicitly activated by entering a valid passphrase before the batch system starts it's operation. The important thing is that by default all tokens are identical. All keys are symbolic links to the CA key. Usually you can activate the batch token and all thinks are ok except that you want to use keybackup or certificate creation. If you enter all passphrases then all tokens are active and you have nothing todo - until the complete system crashs.

7. Performance

7.1. PIII 850MHz, 256 MB RAM

1.000 users, each with one process key_1. My machine swapped sometimes because KDE3, Mozilla, PostgreSQL, Apache and OpenOffice is too much for my old notebook.

Table 16.2. 1000 User test

Function	Time	CA-Token
create_pin	2:39 min	off
check_pin	30:48 min	off (computer swapped)
check_key_params	1:03 min	off
create_key	52:39 min	off
check_key	32:51 min	off (computer swapped)
backup_key	31:47 min	off
check_csr_params	1:12 min	off
create_csr	min	off
complete_csr	min	off
check_csr	min	off
create_cert	min	on
total	:: h	-

Chapter 17. Packaging

1. Common Notices

1.1. Required Perl modules

- XML::Twig
- MIME::Tools
- Net::LDAP - perl-ldap
- CGI
- CGI::Session
- X500::DN
- Net::Server
- Digest::MD5
- Unix::Syslog
- Net::Syslog
- Sys::Syslog
- Time::Local
-

2. RPM-based system

2.1. RedHat/Feodora

2.2. SuSE

2.2.1. HOWTO

1. Copy the directory `suse` to the source directory
2. Go to the SuSE area
3. Build the packages

Example 17.1. SuSE packaging

```
cvs checkout -P openca-0.9
cvs checkout -P suse
mkdir openca-0.9/suse
cp -r suse/* openca-0.9/suse/
cd openca-0.9/suse
make RPMS
cp rpms/* srpms/* your_rpm_directory/
```

2.2.2. Dependancy checking

The biggest problems during packaging are the completeness and the conflicts. You can test the completeness really easy by testing the software itself. The conflicts you can only test by installing all the packages one by one. Please do this with only a basic installation of SuSE to avoid missing or wrong dependencies especially in openca-common. Please install the packages in the following order:

1. openca-doc
2. perl-CGI-Session
3. perl-XML-Twig
4. perl-MIME-Tools
5. perl-Net-Server
6. perl-ldap
7. perl-openca-configuration
8. perl-openca-xml-cache
9. openca-sv
10. perl-openca-openssl
11. perl-openca-crypto
12. perl-openca-tools
13. perl-openca-tristatecgi
14. perl-openca-session
15. perl-openca-pkcs7
16. perl-openca-log
17. perl-openca-ui-html

18. perl-openca-ac
19. perl-openca-req
20. perl-openca-x509
21. perl-openca-crl
22. perl-openca-db
23. perl-openca-dbi
24. perl-openca-common
25. openca-web-interface-node
26. perl-openca-statemachine
27. openca-web-interface-ca
28. openca-web-interface-ra
29. openca-web-interface-pub
30. openca-web-interface-ldap
31. openca-scep
32. openca-web-interface-scep
33. openca-ocspd

3. Debian

4. BSD

Chapter 18. Software Design (legacy from design guide)

1. Database(s)
2. Interface construction
3. openca.cgi
4. libraries
5. modules
6. commands
7. Dataexchange and Node management

Appendix A. History

Here you can read some historical notes about OpenCA. Please don't believe they are complete or always rational. We only write such notes if we have no new ideas at the moment ;-D

1. PKI Scenario before OpenCA

When OpenCA was just a thought in our mind, the need for a complete PKI solution was already perceiving. The problem, in 1998, was that only few products were available in the scenario and the costs associated with this kind of software was indeed considerable. Most of the PKI software available presented non only payment for the software license, but often there were per-certificate costs which prevented many organization to deploy their PKI.

What were the available software then? Not many, indeed most of the implemented software was developed in USA and, because of rigid export regulations, there were problems associated with the adoption of "strong" cryptography outside United States.

Netscape Company was one of the most advanced provider for this kind of software while other providers were pointing their attention on the selling of certification services instead of software capable of managing a PKI. Some other solutions were available from different vendors in Europe too, but the costs problems remained.

Another example of software provider was Secude. The SecuDE, a german security related software company, was selling a crypto layer for generating certificates and maintaining a PKI without limitations on the key-length. Despite of the fact that this software was not imposing weak keys, the company sold licenses based on the number of issued certificates per year.

Let us make a practical example. Let us pretend a local Public Administration, i.e. a small municipality like Modena or one quite big organization, was about to deploy certification services for its citizens. In this scenario it was logical to be prepared for issuing 50,000 - 100,000 certificates (but it should be considered the worst case where every citizen asked for his/her certificate). The number of certificates and the costs related was so high that no one would ever even considered the chance to deploy the services.

Hence the costs problem was one of the most relevant.

All these problems were present not only in PKI managing software, but in crypto enabled applications too. One noticeable example was the popular Netscape browser (and its competitor Internet Explorer), which was available worldwide only with 56-bit capabilities for symmetric crypto and 512-bit for asymmetric crypto. As we underlined, the need of strong cryptography was very high and that is probably the main reason for the availability of products like Fortify that enabled strong cryptography into Netscape.

The Italian scenario presented some points of excellence, though. Two of these, the Modena's municipality and the Turin's Politecnico, were successfully experimenting and developing software based on semi-proprietary solutions (using crypto libraries like the SecuDE one together with ad-hoc developed software). In this highly experimental phases there were some special form of licensing available from different companies (SecuDE was one of the first but in the next few years other research project have been carried out without challenging the real costs because of specific marketing policies by the issuing companies like, for example, Italian Telecom, BNL Multiservizi and others) that were not tied to a per-certificate basis. But these solutions were only temporary and not viable in a real environment with hundreds of thousands of certificates issued and renewed per year.

The natural choice was, then, to look for some freely available software (better if open source so we could modify to fulfill our needs). The only library which was stable enough and available for free was the SSLeay, a collection of libraries and tools for implementing the SSL protocol with some certification capabilities. The SSLeay from Eric Young was already a widely adopted library (indeed nearly all SSL implementing software and SSH solutions on

UNiX already were using it) and it was developed under a BSD like license: this was the solution we were looking for although there was much work to do.

2. PKI and eGovernment

The use of Public Key Infrastructure (PKI) deserves special attention, as the most common challenges for eGovernment to overcome were identified as verifying identity on-line and verifying the authenticity of a document. PKI can address both. The consensus seemed to be that the top level of PKI certificates could be provided like a passport or birth certificate: government defines the format and manages the registry and then provides it via public or private sector, whichever works (like physical passports are delivered).

Some special care, thus, should be taken when addressing PKI's issues as it seems PKI technology can find some very important application in the eGovernment sector. We find interesting that some excellent projects have been carried out using OpenCA in the eGovernment field.

3. Internet Standards

Our project deals mainly with informations and informations need to be interoperable to be useful. Standards are needed in such an environment.

The IETF organization cover every aspect related to the Internet Standards. But what is the IETF? The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.

4. The Project's Purposes

The OpenCA project has been started when PKI's scenario was lacking either on the applications side and on the certificates' management one. We thought that certification technology, although at hand because of the increasing computational computer power and ever decreasing costs, was suffering by lack of open sourced software easing development of digital certificates' related applications and deployment of users acquaintance with digital certificates. The OpenCA Project was, and still it is, a :

> "collaborative effort aimed to develop a robust, full-featured and Open Source out-of-the-box Certification Authority implementing the most used protocols with full-strength cryptography world-wide. OpenCA is based on many Open-Source Projects. Among the supported software is OpenLDAP, OpenSSL, Apache Project, Apache mod_ssl."

The project development has been divided in two main tasks: studying and refining the security scheme that guarantees the best model to be used in a CA and developing software to easily setup and manage the Certification Authority.

One part of the project was the development of a full PKI Open-Sourced software. The software is actually ever growing as new protocols and standards are being issued by the standardization bodies. The users' and developers' community has been enlarging and we can now count on many individuals and organizations directly or indirectly supporting the project.

Although the software development is one important aspect of OpenCA, our objectives were not only technological. We can say that among the project's objectives there were also the exchange of ideas between different realities and the setting up of a working group deploying code and solutions solving practical certification issues.

5. The Project's Achievements

Five years have passed since we started to work at the project. OpenCA has been constantly growing and the number and quality of active installations are, today, too many to be mentioned here. We have to state, for clarity, that the project is still evolving and because of this we can only imagine what the future of the project will be. Today the project counts a quite high number of developers, mainly from universities, and contributing users.

The basic principles that have been followed throughout these development years:

- adherence to the IETF standards - at every step we tried to stick to the RFCs covering the matter from the very beginning. Indeed the project basic structure is derived from one IETF specification.
- feedback from users - the project have grown integrating help and keeping in mind objections by users and developers.
- openness - we tried to keep our sources as readable as possible thus allowing almost everyone to contribute with code or comments.
- interoperability - this is a consequence of being adherent to the standard but not only. Whenever possible we tried to support the majority of the systems today available. For example we introduced the usage of Java applets to support Microsoft Internet Explorer and we develop support for SCEP.
- usage of simple programming language (whenever possible) - this follows from the consideration that security is not obscurity and we needed to be as readable as possible. Then we choose for PERL which guaranteed for portability and simplicity.

Thus most of the project's objectives have been either achieved and yet are still to be achieved. This could be quite a cryptic sentence but it is not. The project has achieved its goals because it helped in searching for solutions about certification problems and issues. For example OpenCA has made possible for some municipalities to deploy services to its citizens and other organizations to develop applications for their customers, we solved many practical problems together with users and developers and we helped in making this technology available for almost everyone. We also contributed to the discussion on the pkix working group about some specific issues like certificates' suspension. Still the project has not yet achieved its goals because much work is needed for this technology to be widely deployed and accepted by users. Many standards and protocols are engineered every day and thus the project still needs active development to be kept up-to-dated.

6. The OpenCA Project

6.1. The project start

We started the OpenCA project in June 1998. The very original idea was developed by Massimiliano Pala. The first version of OpenCA has been developed in approximately one monthand the project first sources were composed of a single quite long script. When the first version of the software was developed the OpenSSL project was still named SSLeay and it was very different from what it is now: many functions were still buggy and many others were missing.

Installation of the project was based on a very simple Makefile and some scripts to initialize the CA. To quickly install the software you just needed to unpack the archive, cd to the newly created directory, use the 'make install' command: a script was then run to install the base CA software and generate (and eventually sign) the CA certificate.

A series of script was provided to help in installing and configuring most of the project's parts. Although very simple, this solution caused many problem with the users' community because of issues rising from the need of being able to better customize the full package. For example to be able to generate certificates through the full featured process you needed to install the CA web, RA server software and generate RAs' browser importable certificates. The first version

of OpenCA was very basic, functions implemented were aimed primarily to issue certificates and CRLs only and the installation method was quite rude: no usage of any configuration utility (i.e. autoconf and automake) and scripts were compatible with bash only.

New releases were adding ever more features to the project and thus the 0.109a version was already including interfaces for either CA, RAServer and Public. From the very beginning of the project and since the release of the first version a great attention from the Internet community was turned on the project.

6.2. Offering Help to Other Projects: OpenSSL

Our project has been based on the OpenSSL cryptographic tool. The OpenSSL project was named SSLeay and has been developed originally by Eric Young and then it has been taken in charge by Ralf S. Engelschall and recently by Steve Henson. Although still one of the most advanced (if not the only one in 1998) open source crypto library available, we needed many features not available in the toolkit at the time we started OpenCA. The availability of the sources has been proved essential for us because we have been able to add many new features no one have thought about before. This is not a minus for OpenSSL. This is a plus for open source and OpenSSL will be extended day by day by other developers who also need additional things.

All this work had permitted us to have the needed functions to correctly manage certificates and requests by building a web oriented interface acting as a front end to the crypto layer.

First versions of OpenCA were strictly tied to OpenSSL but we knew we had to abstract from the crypto library as much as possible. If we could be able to have a layered structure it could help external developers to support different cryptographic layers and thus incrementing the project's portability. This has been done since version 0.6.0 which had many new features and a much more complex structure then the former releases.

Because of the constant growth of the software developed and the asking for direct support by the community, we choose to move for advanced development tools. Many people have found themselves to work on different aspects of PKIs and in particular on the Project features and structure.

6.3. CVS and Mailing Lists

> "CVS is a version control system, which allows you to keep old versions of files (usually source code), keep a log of who, when, and why changes occurred, etc., like RCS or SCCS."

We used CVS because it helped to manage releases and to control the concurrent editing of source files among multiple authors as new developers were admitted to directly access the main source code.

Thus we needed to install it on our main server, which acted as either package download site and as community support site via the installed web server and different applications: a CVS server and a mailing list management tool. Some difficulties were found for the CVS setup and management because of many security issues we had to care about. Anyway these tools let us work in collaboration with many different developers and contributors from all around the world with little administration efforts.

6.4. The Open Source Choice

Security is not obscurity. This is a principle we should all keep in mind when approaching the security issue.

We can therefore state that the Open Source choice, especially when dealing with security and issues tied to this subject, is important because it gives the chance to everyone to study and test our ideas and eventually help in designing a better solution. If we take for true that two eyes are better than one, then why not one hundred or one thousand?

Open Source Projects have indeed another one big advantage: contribution can come directly from users and/or developers who need features (or modifies) you simply have not had the time to think about or to implement. We found that Open Sourcing the Project have given a very big help in growing either in its practical implementation and in its visibility.

6.5. Migrating to SourceForge

We found ourselves at the end of the 2000 with the need to have more administration power, to coordinate different developers, and Internet connectivity. We have been asked to move on different servers by the users and proposals were not missing. Anyway we decided to move to SourceForge. At the moment of this writing we have many mirrors around the world updated daily, mainly thanks to the Sunsite and SourceForge networks, giving us much visibility over the whole Internet. This helped especially Asian users that suffered many times from lousy connections and long download times.

The SourceForge was then the natural choice where to move the project core to. This has saved much administration work and thus has let us work much on the project than on the development tools used.

Appendix B. Authors and Contributors

Here you can find all authors and contributors of this guide.

1. Michael Bell

Design and all content which is not explicitly written by others

2. Chris Covell

LDAP stuff and Web interface documentation

3. Massimiliano Pala

History

4. Ulrich Bathels

Sendmail configuration for basic SMTP authentication []

5. FAQ

Several people help us on the mailinglists. We extract many hints and workarounds from these lists and add them to the FAQ. We try to list here the guys to help us. If we forget somebody please help us and send a notice.

- Dejan Kulpinski - Microsoft Smartcard Logon
- Gottfried Scheckenbach - Microsoft Outlook Express

Appendix C. FAQ

1. General PKI Issues

1.1. What is a certificate?

A certificate is a so called digital ID card. The correctness of a certificate will be guarnteed by a certificate from a higher level of the hierarchy. Such a certificate is called CA certificate.

1.2. Which informations does a certificate contain?

Certificate Informations

- serial number of the certificate
- a subject (name)
- the corresponding public key to the private key of the certificate owner
- the name of the issuer
- the version of the certificate
- the used cryptoalgorithms to create the certificate
- the validity period
- some extensions
- the digital signature of the certificate

1.3. What is a request?

There are two types of requests CSRs and CRRs. CSRs are used to ask a trustcenter for a certificate. CRR are used to ask a trustcenter to revoke a certificate if it is corrupted. There are two important standards for CSRs - PKCS#10 and SPKAC. OpenCA can handle both standards automatically.

1.4. Which information does a CSR contain?

CSR Informations

- a subject (name)
- the version of the request
- the corresponding public key to the private key of the certificate owner
- some attributes

1.5. What is a CA?

1.6. Why should I not place the CA on the same machine like the RA?

1.7. What is an extensions?

1.8. I use Windows 2000 and Internet Explorer 6 SP1 and it don't show any CSPs.

It is fairly well known that there are two versions of Xenroll.dll used by versions of IE to create certificate requests and manage CSPs etc.. OpenCA since version 0.9.1 has managed them via the ieCSR.vbs scripts.

We have noticed that if a user has Win2K and IE6 SP1 then the version of xenroll.dll does not work and the users can see no CSPs to manage their certs with. A patch is required from Microsoft (323172) for Win2k, or it needs to go up to SP3. You can host a copy of the latest xenroll.dll on your web site under a CertControl directory and it will be downloaded and installed automatically.

As far as we can tell, the latest xenroll.dll is a different file, but shares the same identifiers as the pre-patched version. We have noticed that the isCSR.vbs (as of 0.9.1) is written in such a way as to not expect there to be a non working version of xenroll.dll, so there is a bit of a gap.

1.9. How can I setup a sub CA?

1. Initialize the SubCA (initialize database, generate secret key, generate request)
2. export request
3. untar the export (to get the careq.pem), the next steps are only correct if you use OpenCA for your Root CA
4. Point to the Root CA public interface -> request a certificate -> server request -> browse for the careq.pem and submit the request
5. Point to the Root CA RA interface and approuve the request, upload to the Root CA CA; point to CA interface, issue the certificate
6. Download the certificate for the sub CA via the RA or public interface of the Root CA
7. rename the file to cacert.pem and manually make a new tar
8. Point your browser to the SubCA CA interface and import CA certificate approuved by Root CA

2. General OpenCA Issues

2.1. Does it be possible to revoke a certificate without any user interaction?

Yes, it is possible. Go to a RA interface. Go to the certificate which you want to revoke. View the certificate. Click on revoke, fill out the form and now you have created the initial CRR to revoke the certificate.

2.2. I try to add a role and get the message “The role XYZ exists already!”

This message appears if one of the configurationfiles of the new role already exist. Please check the files in the directories `OPENCADIR/etc/openssl/extfiles` and `OPENCADIR/etc/openssl/openssl`.

2.3. All cryptographic operations fail.

Check that the configuration option `OPENSSL` is set to the correct path. It mus be the binary of OpenSSL. You have to verify all files in `OPENCADIR/etc/servers/`.

2.4. Apache's error_log reports a nonexistent option `-subj` of openssl req

You are using OpenSSL 0.9.6 but you must use 0.9.7. The use of 0.9.6 can cause inconsistent data. Normally OpenCA cannot installed if OpenSSL 0.9.7 is not present. So please check the path to the OpenSSL binary in the configuration files. The option is `OPENSSL` in all files in `OPENCADIR/etc/servers/`.

2.5. Apache's error_log contains a message from IBM DB2 that the environment is not setted

Please check the settings in `etc/servers/DBI.conf` because this happens if IBM's software cannot find the libraries and databases.

2.6. What do the new features of 0.9.2 be?

- it is now possible to create usable packages
- you can configure the PKI after the installation
- docbook based documentation
- integrated access control
- secure export of private keys via the public interface
- several LDAP improvements
- keysizes are now choosable for IE users too
- much better CSR editing
- additional attributes for requests (e.g. telephonenumbers)
- menugeneration via XML-configurationfile
- SCEP support
- warn expiring certificates
- more (an explicit) download formats for certificates

- subject verification for PKCS#10 requests
- logging support

2.7. I try to approve and sign a request with Mozilla and it fails.

Mozilla doesn't implement crypto.signForm or anything else to sign HTML forms. There is a really good patch or better bugfix for Mozilla but they don't include it into the releases. You can find a patched Mozilla at WaMCom.org [http://wamcom.org/]. The patch can be visited at Mozilla [http://bugzilla.mozilla.org/show_bug.cgi?id=29152]. The bug ID is 29152. We don't know why it takes so long to fix such a small security problem but until now we can only recommend to use the Mozilla version from WaMCom.org.

There is a second fix for this problem secclab. This is plugin which is available from mozdev.org but until now it is not stable enough to be supported from us. If it is stable and Mozilla still doesn't support crypto.signFrom then we will support secclab.

2.8. I try to approve and sign a request with Konqueror (KDE) and it fails.

KDE doesn't include any functionality to sign HTML forms until know. So this feature is not supported for KDE.

2.9. How is the format of the disc to import the CA certificate from the root CA?

It is a noncompressed tar file. The name of the file which contains the CA certificate is cacert.pem. The format of the file is PEM (sometimes called CRT or base64 encoded).

2.10. OpenSSL reports entry 1: invalid expiry date

If you try to create a CRL, to issue a certificate or to revoke a certificate and it fails then you should get an errormessage from OpenSSL. If the errormessage include the string entry 1: invalid expiry date then the database file `index.txt` is damaged. The easiest solution is to go to the backup and recovery are of the node management interface. There you can use the link which starts the rebuilding of the OpenSSL files. After this operation the OpenSSL files are correct again.

2.11. Outlook cannot encrypt mail with imported certificate

If you imported the certificate of another user and try to send him an encrypted email then it can happen that this doesn't work with Outlook and Outlook Express. The reason is that the person must be present in your contacts. The best way to add the person to your contacts is to take a signed email and import the user from this email to your contacts.

2.12. My Outlook freezes after I received a signed email

There are several events why Outlook freezes but one events is a signed email in combination with an anti virus program. One user reports some time ago a frozen Outlook in combination with an anti virus program from Kapersky. Like often with Microsoft programs it is not clear why Outlook crashs and who makes the mistake and includes a bug in it's program.

2.13. General Error 6751 during certificate issuing

If you try to issue a certificate and you use an OpenCA version prior to 0.9.2 then it is possible that you get a general error 6751.

Example C.1. General error 6751 during certificate issueing

```
Error 6751
General Error. Error while issuing Certificate to CA Services some.host.com
 (filename: /usr/local/openca/var/tmp/04.req).

OpenCA::OpenSSL returns errocode 7731071
 (OpenCA::OpenSSL->issueCert: OpenSSL fails (256).)..
```

If you check your Apache's `error_log` then should see some lines which include digital envelope routines:EVP_DecryptFinal:bad.

Example C.2. Bad passphrase error log during certificate issueing

```
[Mon Dec 29 18:32:59 2003] [error] [client 192.168.1.38]
        unable to load CA private key, referer:
        http://ca.localhosts.com/cgi-bin/ca/ca?cmd=viewCSR;dataType=APPROVED_REQUEST;key=1
[Mon Dec 29 18:32:59 2003] [error] [client 192.168.1.38]
        18685:error:06065064:digital envelope routines:
                EVP_DecryptFinal:bad decrypt:evp_enc.c:438:, referer:
        http://ca.localhosts.com/cgi-bin/ca/ca?cmd=viewCSR;dataType=APPROVED_REQUEST;key=1
[Mon Dec 29 18:32:59 2003] [error] [client 192.168.1.38]
        18685:error:0906A065:PEM routines:
                PEM_do_header:bad decrypt:pem_lib.c:421:, referer:
        http://ca.localhosts.com/cgi-bin/ca/ca?cmd=viewCSR;dataType=APPROVED_REQUEST;key=1
```

The reason is very simple. The messages unable to load CA private key and EVP_DecryptFinal:bad decrypt are from OpenSSL and signal that the CA's private key cannot be decrypted. This usually happens if you use a wrong passphrase. You can test your passphrase with the command **openssl rsa -in /usr/local/openca/var/crypto/keys/cakey.pem -text -noout**. If it fails then your passphrase is wrong.

3. Installation Issues

3.1. FreeBSD and OpenCA

3.1.1. make

FreeBSD and potentially several other BSDs use an own make. Nearly the complete development of OpenCA is done on Linux. The result is that our makefiles works perfectly with GNU make but there are several unresolved problems with FreeBSD because we cannot test and fix on FreeBSD. The solution is to add GNU make to configure. You can do this this way:

Configure with GNU make

```
MAKE=/usr/local/bin/gmake ./configure --your-options
```

Please check `Makefile.global_vars` to include the correct setting of `MAKE`. If the correct setting of make is missing then please add the following line:

Set GNU make in `Makefile.global_vars`

```
MAKE=/usr/local/bin/gmake
```

GNU make is fully tested with OpenCA.

3.1.2. install

There is an additional problem with install like with make. FreeBSD doesn't support the parameter `-D`. This parameter is only required if some of the used installation paths like `BINDIR` do not exist. If you perform a normal installation and use default paths then there should be no problems without `-D`.

You have to check `Makefile.global_vars.in` before you run ./configure or `Makefile.global_vars` after you run ./configure. There you have to remove the parameter `-D` from the definition of the variable `INSTALL`.

If you are not the hacker which like to dig for bugs and the removal of `-D` doesn't work then you have one final option. You can install the port "coreutils" which includes the GNU install. this is not a real bugfix but it works.

3.2. What is a hierarchy level?

OpenCA includes some protection mechanism to avoid state injections during the dataexchange. Therefore you have to configure the level at which OpenCA performs the dataexchange, so that **./configure** can make a good preconfiguration. This is only correct for 0.9.1. 0.9.2+ uses `OPENCADIR/etc/config.xml` to configure the dataexchange.

1. If you export from a public area to a server with the RA then must use `pub`.
2. If you export from a server with a RA to a server with a CA then you have to use `ra`.
3. If you export from a server with a RA to a server with a public interface then you have to use `ra`.
4. If you export from a server with a CA to a server with a RA interface then you have to use `ca`.
5. If you have another case then you have to chosse one of the above defined options and then you have to edit the options for the dataexchange manually like described in the administrator guide.

3.3. Undefined subroutine &main::xyz

Usually the errortext is a little bit more descriptive and looks like this:

Example C.3. Full errormessage for missing functions

```
[Thu Oct 09 14:50:52 2003] [error] [client 127.0.0.1] Undefined
subroutine &main::configError called at /var/www/cgi-bin/ca/ca line 86.,
referer: http://localhost/htdocs/ca/
```

The reason for such an error is a missing library - in this case `misc-utils.lib`. There is only one well-known error - you used `--enable-package-build` as configure option. This happens if somebody uses a configure example which is used for package builds. The disables the installation of all common software parts like modules, libraries and configuration files. If you configure again without `--enable-package-build` and then run make install-xyz again then all libraries should be present.

3.4. Symbolic link installaton failed

Several user observed that OpenCA 0.9.1.x installation failed during the installation of symbolic links for commands. The errormessage simply reports an already present symbolic link.

Example C.4. Already present symbolic link

```
 make[8]: Entering directory
`/home/linux/tar/openca-0.9.1.2/src/web-interfaces/ca/cmds'
cd /usr/local/openca.0.9.2/openca/lib/servers/ca/cmds; \
        ln -s ../../../cmds/add_module
ln: `./add_module': File exists
make[8]: *** [/usr/local/openca.0.9.2/openca/lib/cmds/add_module] Error 1
make[8]: Leaving directory
...
make[1]: Leaving directory `/home/linux/tar/openca-0.9.1.2'
make: *** [install-ca] Error 2
```

The reason for this failure is a defect symbolic link. The makefile removes an already present symbolic link only if the symbolic link is correct. If the target of the symbolic link doesn't exist then the symbolic link is not removed and the error occurs.

3.5. After the installation all common parts are missing

If the command **make install-*** doesn't report an error and after the installation all common parts like modules, configuration, libraries, openca-sv and commands are missing then please check your configure options. Sometimes the people simply copying our configuration examples without noticing that some of these examples are used for package creation. These examples include an option `--enable-package-build`. This option prevents all common stuff from installation to build better packages.

3.6. Conflicting Modules

Sometimes there are reports about errormessages which signal an unknown login type or another empty configuration option. First you should check that there are at minimum two running processes - one is the OpenCA daemon which handles the requests and one is the XML cache. If there is only one server then there is a problem during the startup of the daemons. There two well known issues during the startup - wrong permissions and duplicate installations of a module. Please allways check first that there are no problems with the permissions.

If you are sure that you have no problem with the permissions then it is usually a problem with different installed version of the Perlmodule XML::Twig. This can cause a crash of the daemon at startup or always empty results for a configurationfile. First you should search for the different version of XML::Twig.

Example C.5. Search for XML::Twig modules

```
## every Unix
find / -name Twig.pm -print

## rpm based Unix (RedHat, SuSE, Mandrake ...)
rpm -qa | grep -i twig

## dpkg based Unix (Debian)
dpkg -l "*twig*"
```

After you know the installed modules you have to remove all versions older than 3.09. OpenCA was developed with 3.08 and 3.09 but the last tests were only performed with 3.09. So we don't know what's happen if you use an earlier version.

3.7. The xml path to the access control is missing

If you installed OpenCA and try to login to OpenCA then it can happen that you see the errormessage The xml path to the access control is missing. The errorcode should be 6292010. Usually the missing XML path is `access_control/acl_config/module_id`.

Please check that a XML parser is installed on your system. We use XML::Twig to parse huge XML files. This module uses an already installed XML parser. If there is no XML parser installed for XML::Twig then we cannot read and interprete our configuration files.

3.8. Unknown Login Type

Sometimes you get the errormessage An unknown login type was specified in the configuration! after you installed OpenCA and try to login for the first time. The errorcode in this situation should be 6273966.

Please check the UID of the socket which the XML cache uses. Sometimes the UID of the socket is root and not the UID of the webserver. We don't known how this can happen. Please verify that this socket is not owned by root.

To solve the problem you can stop openca with **openca_stop** or **openca_rc stop**. Then you have to remove the socket `OPENCADIR/var/tmp/openca_xml_cache`. After this you can start openca again and please verify that the permissions and owner are now correct.

3.9. Type Mismatch during request generation with Internet Explorer

If an Internet Explorer reports a type mismatch during the request generation then please check the configuration of your Apache. Usually the reference to the VB script `ieCSR.vbs` is wrong. This happens if the document root of the Apache is wrong.

Example C.6. Type Mismatch on Internet Explorer

```
Line: 88
Char: 1
Error: Type Mismatch: 'GenReq'
Code: 0
URL://xxxxxxxxxxxxxxx/cgi-bin/pub/pki?cmd=basic_csr
```

4. Configuration Issues

4.1. How can I configure my httpd.conf for virtual hosts?

Here is a small example for the configuration of a virtual host for OpenCA.

Example C.7. virtual host configuration

```
<VirtualHost _default_:443>
      ServerName 157.159.100.42:443
      ServerAdmin pascal.verrecchia@int-evry.fr
      DocumentRoot /srv/ra/apache/htdocs
      ErrorLog /usr/local/apache/logs/error_log
      Options MultiViews Indexes Fol ..............
...........

</VirtualHost>
```

It is important to bind the address and the port this mean that you should include the following statement in you httpd.conf:

```
BindAddress Your_address_IP:80
Listen 80
```

You can also add `BindAddress *` to be sure.

4.2. How can I configure virtual hosts with ./configure?

Here is a small example from an OpenCA user which the developers never planned but it works.

Example C.8. ./configure and virtual hosts

```
--with-ca-htdocs-url-prefix=http://ca.dskt6807.zhwin.ch \
--with-node-htdocs-url-prefix=http://node.dskt6807.zhwin.ch \
--with-ra-htdocs-url-prefix=http://ra.dskt6807.zhwin.ch \
--with-ldap-htdocs-url-prefix=http://ldap.dskt6807.zhwin.ch \
--with-pub-htdocs-url-prefix=http://pub.dskt6807.zhwin.ch \
```

The example is from OpenCA 0.9.1. Please do these configuration in `config.xml` if you use OpenCA 0.9.2 or later.

4.3. I have some users which should not be published in LDAP. Does it be possible with OpenCA?

Yes, it is possible. There is an option `LDAPexcludedRoles` in the configuration files of the node and the ldap interface. If you add a role there then all certificates which have this role will not be published via the LDAP server.

4.4. Does it be possible to authenticate users by their certificates at the apache before they will be authenticated by OpenCA itself?

Yes, you can edit the httpd.conf in the appropriate way. Please don't do this for the public interface.

Example C.9. Client authentication with mod_ssl

```
<VirtualHost ra.mycompany.de:4443>

   ServerName ra.mycompany.de
   DocumentRoot /RA/apache/htdocs
   ServerAdmin nicolaie.szabadkai@mycompany.de
   SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
downgrade-1.0 force-response-1.0

   SSLEngine on
   SSLCertificateFile          /RA/ssl.crt/server.pem
   SSLCertificateKeyFile       /RA/ssl.key/key.pem
   SSLCertificateChainFile     /RA/OpenCA/var/crypto/chain/cacert.crt
   SSLCACertificateFile        /RA/OpenCA/var/crypto/cacerts/cacert.pem
   SSLCARevocationFile         /RA/OpenCA/var/crypto/crls/cacrl.pem
   SSLVerifyClient require
   SSLVerifyDepth  10
   SSLOptions +StdEnvVars +ExportCertData +StrictRequire

   ErrorLog /var/log/httpd/ra.srv.err.log
   CustomLog /var/log/httpd/ra.srv.req.log "%t %h %{SSL_PROTOCOL}x
%{SSL_CIPHER}x \"%r\" %b"

   ScriptAlias "/cgi-bin/" "/RA/apache/cgi-bin/"
   <Directory "/RA/apache/cgi-bin">
       AllowOverride None
       Options FollowSymLinks
       Order deny,allow
       Deny from all
       Allow from 10.1.114 10.100.1 10.1.102
       SSLRequireSSL
       SSLRequire ( %{SSL_CLIENT_S_DN_O} eq "MyCompany" \
               && %{SSL_CLIENT_S_DN_CN} =~ m/ramanager?/ )
   </Directory>

   ...
</VirtualHost>
```

If you no go to the RA then you have to choose a certificate which your browser will use to authenticate you to the Apache.

4.5. I want to update my 0.9.2 installation. Is this dangerous?

It is necessary that you update your OpenCA installation from time to time. This can happen if there is a new security advisory or some normal bugs are fixed. Since OpenCA 0.9.1 there are bugfix releases. These releases only update the software. They never touch the configuration. If you have a 0.9.1.3 and you have to update to 0.9.1.4 then simply use the same configuration like for 0.9.1.3. The installation with make never overwrites the `etc` or `var` area of your already existing installation. Nevertheless it is strongly recommended to backup your complete installation before you start such critical operations like an update.

4.6. I want update to 0.9.2. How can I update my sql database?

There are two general methods. You can use backup/restore or you add the columns by hand.

If you want to use the first possibility then you have to do the following:

1. Create a backup from the database with the node interface.
2. Remove all tables and sequence generators.
3. Update OpenCA.
4. Go to the node interface.
5. Initialize the database.
6. Go to the recovery page of OpenCA.
7. Import the database.
8. Restore OpenSSL's files.

If you want to update the database by hand then you have to do the following:

1. Login to the database.
2.
```
alter table request add scep_tid TEXTTYPE;
alter table log add scep_tid TEXTTYPE;
alter table request add loa TEXTTYPE;
alter table certificate add loa TEXTTYPE;
alter table crr add loa TEXTTYPE;
alter table log add loa TEXTTYPE;
```

   TEXTTYPE differs for every database. The following table contains the correct type names for every supported database.

Table C.1. Texttypes for different databases

Database	type
mysql	TEXT
Pg	text
DB2	long varchar
Oracle	varchar2 (1999)

3. Go to the node interface.
4. Administration.
5. Databasehandling.
6. Update searchable attributes.

4.7. If I run openca-ocspd then I obtain a segmentation fault.

Some releases include an incomplete sample config. You have to add something like this to your ocspd.conf:

Example C.10. OCSP configuration for LDAP

```
[ OCSPD_default ]
....
dbms  = ocsp_crl

[ ocsp_crl ]
crl_url = ldap://my.ldap.server
crl_entry_dn = cn=MyCA,ou=CA,o=MyOrg,c=MyCountry
```

Alternatively you can use http too:

Example C.11. OCSP configuration for http

```
[ ocsp_crl ]
crl_url = http://my.ca-public.server
crl_entry_dn = /crl/cacrl.crl
```

4.8. I installed a second public interface, run configure_etc.sh and now are all the paths in the other public interface wrong.

Before you run configure with the changed config.xml for the second public interface you have to reduce the scope of the files in **configure_etc.sh** to the new interface.

After such a crash you can configure `config.xml` to the old values, set the paths in `configure_etc.sh` to the first interface only (!!!) and then run **configure_etc.sh** again.

4.9. I issue a certificate for a mailserver but sendmail doesn't work and reports an errormessage which includes “reason=unsupported certificate purpose”

Please read the notices about SMTP servers in the OpenSSL section of the administration guide. If you only have one certificate for your mailserver then it must include the extensions for SSL servers and SSL clients. The extensions for SSL servers are not enough because SMTP servers act as clients too.

4.10. My (Microsoft) client hangs after it tries to start a secured connection

There are some situations where clients hang after they try to connect to a TLS or SSL secured server. Examples are Microsoft Outlook clients which connect to mail servers which use TLS or Microsoft Internet Explorers which try to connect to a https server.

Usually the certificate contains a CRL distribution point (CDP) which uses https or ldaps as protocol. The result is that the client tries to verify the server certificate and opens a connection to the server which stores the CDP. If this server presents a certificate which contains a CDP with TLS protection then you have a perfect loop. This can also happen if you try to verify a client certificate which includes a TLS or SSL secured CRL distribution point.

There are two solution for this problem. First you can use only http and ldap or other supported protocols for CRL distribution which don't use TLS and SSL. This is not a big security risc because CRLs are protected by the signature of the corresponding CA. Second you use https or ldaps for client certificates but http or ldap for server certificates. This will produce only one loop if the server certificate will be verified.

If you already enrolled an infrastructure and now you are running into problems with hunderds or thousands of client certificates then you should use the second option to solve your problems. If you enroll new certificates for the servers then you have no problems with your endusers - you have not to explain the problems, the installation of new certificates and the reasons why you don't expect such problems. You “only” install some new server certificates and all problems are fixed like a simple network problem.

4.11. Outlook freezes when receiving a signed Mail but worked already fine for some days

The CDP of the certificate from the signature points to a SSL-secured website which was signed by the same CA than the mail certificate. Best solution: Change the CDP to non-https url or a https-url signed by another CA and reissue the mail certificates. If you dont want to reissue all your mail certs it's ok to just change the webservers CDP URL and reissue the webserver certificate.

4.12. During the request generation OpenCA fails and reports a too short textfield

Old versions of OpenCA include a hardwired minimum length for HTML-textfields. The minimum length was three. You can change this limit in `basic_csr`. New versions of OpenCA can be configured. Please read the “installation and configuration guide” Section 4, “CSRs”.

4.13. Can I place my organization's logo on the web interface?

Yes, please check `etc/config.xml`. There are two options `menu_logo_left` and `menu_logo_right` which can be used to place logos in the menuframe. Please be careful with this feature because it can reduce the usability of the software.

4.14. Microsoft Smartcard Logon

The original mail was from Dejan Kulpinski which exactly describes for the first time how to establish a Smartcard Logon to Windows 2000 domain using OpenCA certification authority. The mail is present in the mail archives of OpenCA-Users@lists.sf.net.

Warning

Before the original article starts one important note. If you use a TTP certificate for Microsoft domaincontroller then the SMTP based replication of Microsoft doesn't work. Microsoft officially commits this problem but doesn't comment the reasons.

4.14.1. Original Description for 0.9.1

Well the story is pretty long, so I start from the beginning.

Microsoft Knowledge Base Q291010 defines the requirements a DC certificate must have in order to be able to authenticate smartcard logon users. One of them is that key pair must be genarated using RSA/Schannel CSP. The most simple way for doing this is to install Microsoft CA services and to issue a certificate request using a form, where You can choose this CSP and generate a key pair and certificate request in PKCS#10 format.

Then you need to create a role for DC certificate because of specific extensions it must have. The first is the subjecAltName extension that has to have other name = GUID of DC and DNS name = DNS name of DC. Since I have already installed Microsoft CA services, it has generated DC certificate, so the most simple thing for me was to export it to OpenCA machine and parse it with openssl asn1parse utility (I'll send you a detail description if you need) to get a DER formatted subjectAltName. Then I did a hex dump of this DER file and put the result after the "subjectAltName=DER:" in ext file of newly created role. The second is the Certificate Template extension that is specific to Microsoft and unknown to openssl so I didi the same thing with asn1parse and put the result after the "1.3.6.1.4.1.311.20.2=DER:" in ext file (1.3.6.1.4.1.311.20.2 is the OID of Certificate Template extension). You can just use this (but subjectAltName is specific to You so use asn1parse):

OpenSSL configuration for domaincontroller

```
keyUsage = digitalSignature, keyEncipherment

# extended Key Usage
extendedKeyUsage = clientAuth, serverAuth

# Certificate Template "DomainController" (bmp string)
1.3.6.1.4.1.311.20.2=DER:1e:20:00:44:00:6f:00:6d:00:61:00:69:00:6e:00:43
:00:6f:00:6e:00:74:00:72:00:6f:00:6c:00:6c:00:65:00:72
```

That is it.

Next, you have to generate a smartcard user certificate that also must have some specifics (so new role again). Beside the "CRL Distribution Points" extension that is standard, the "Enhanced Key Usage" extension must contain Cleint Authentication and Smartcard Logon. So you have to put something like

extendedKeyUsage for clients

```
extendedKeyUsage = clientAuth, 1.3.6.1.4.1.311.20.2.2, 1.3.6.1.5.5.7.3.4
```

where 1.3.6.1.4.1.311.20.2.2 is the OID of SmartCard logon, and 1.3.6.1.5.5.7.3.4 is the OID of SecureMail (if you want to use this certificate for Secure mail too). Certificate Template extension must have a value "SmartcardUser" or "SmartcardLogon (again, with SmartcardUser you can also use the certificate for SecureMail). You can use the this:

OpenSSL configuration for clients

```
keyUsage = digitalSignature, keyEncipherment

# Certificate template "SmartcardUser" (bmp string)
1.3.6.1.4.1.311.20.2=DER:1e:1a:00:53:00:6d:00:61:00:72:00:74:00:63:00:61
:00:72:00:64:00:55:00:73:00:65:00:72

# Enhanced Key usage (clientAuth, SmartcardLogon, secureMail)
extendedKeyUsage = clientAuth, 1.3.6.1.4.1.311.20.2.2, 1.3.6.1.5.5.7.3.4
```

And the last thing, subjectAltName extension must have other name = Principal name = user_principal_name (for example jsmith@domain.com). To be able to issue the certificates to different smartcard users without the need to create the extension for each, I have used the OpenCA possibility to automatically set user email in subjectAltName (but in this case you have to use jsmith@domain.com as user email, not John.Smith@domain.com because this is not the principal name). Then I put next in smartcard user ext file:

subjectAltName for clients

```
subjectAltName=otherName:1.3.6.1.4.1.311.20.2.3;UTF8:${ENV::subjectAltName}
```

where 1.3.6.1.4.1.311.20.2.3 is Principal Name OID. Remember that is works only with openssl version 0.9.8 (you can download a snapshot of 0.9.8-dev version and use it)

The last problem is that OpenCA automatically adds the word "email:" before the actual email address. To eliminate this I have been told to remove it in `$OPENCA_PREFIX/lib/functions/crypto-utils.lib`

Code fixing in 0.9.1

Replace

```
 $ENV{'subjectAltName'} =
"email:".$req->getParsed()->{DN_HASH}->{EMAILADDRESS}[0];
```

with

```
$ENV{'subjectAltName'} =
$req->getParsed()->{DN_HASH}->{EMAILADDRESS}[0];
```

4.14.2. Additional Notes for 0.9.2

OpenCA 0.9.2 is much more flexible. You don't need to patch the code to include the othername into the subject alternative name. You have to add *"otherName"* to the configuration option `CSR_SUPPORTED_SUBJECT_ALT_NAMES`.

You can find this option in `OPENCADIR/etc/servers/ra.conf`. If you edit the request then you must choose "otherName" as a part of the subject alternative name and the value must be

OpenCA 0.9.2 othername value

```
1.3.6.1.4.1.311.20.2.3;UTF8:your_Microsoft_UPN
```

OpenCA 0.9.2 only need configuration adjustments to work with Microsoft Smartcardlogin if you use OpenSSL 0.9.8.

4.15. Cannot create new OpenCA tokenobject

The correct errormessage is usually:

Cannot create new OpenCA tokenobject

```
Configuration Error.
Cannot initialize cryptographic layer (configurationfile /usr/local/pki/UnicibleICA4/etc/t
Cannot create new OpenCA Tokenobject.
```

Please check the configurationfile `OPENCADIR/etc/token.xml`. The reference to OpenSSL must be correct with the full path and not only `openssl`.

token.xml with correct openssl reference

```
<option>
  <name>SHELL<name>
  <value>/usr/local/ssl/openssl<value>
<option>
```

4.16. How can I use a Luna token with OpenCA 0.9.1

You have to edit two files - OpenCA's `ca.conf` and Chrysalis-ITS's `Chrystoki.conf`. First you should configure OpenCA in `OPENCADIR/etc/servers/ca.conf`.

OPENCADIR/etc/servers/ca.conf

```
## HSM configuration
## =================

## Example: LunaCA3

opensslEngine    "LunaCA3 "
opensslEngineArg ""

HSM_LOGIN_CMD   "/usr/luna/bin/ca3util -o -s 1 -i 11:10"
HSM_LOGOUT_CMD  "/usr/luna/bin/ca3util -c -s 1 -i 11:10"
HSM_GENKEY_CMD  "/usr/luna/bin/ca3util -s 1 -i 11:10 -g @__BITS__@ -f @__OUTFILE__@"
```

After this you must configure the Luna device.

/usr/luna/etc/Chrystoki.conf

```
Chrystoki2 = {
   LibUNIX=/usr/lib/libcrystoki2.so;
}
CardReader = {
  RemoteCommand=1;
}
Luna = {
  DefaultTimeOut=500000;
  PEDTimeout1=100000;
  PEDTimeout2=100000;
}
EngineLunaCA3= {
        LibPath=/usr/luna/lib/libcrystoki2.so;
        EngineInit=1:11:10;
}
```

Now OpenCA 0.9.1 should be ready for Chrysalis-ITS LunaCA 3.

4.17. How can I include a complete certificate chain into a PKCS#12 file?

If you enroll a certificate and a private key to a user via file in PKCS#12 format then you usually want to include the complete certificate chain. This is necessary because many software products doesn't work if the chain is incomplete. This can be normal mail programs or VPN clients. The price is no argument in this case.

Otherwise there can be problems if you try to install certificates which are already present at the target system. The worst case is the destruction of already exeisting certificate chains by overwriting an old CA certificate. Therefore

OpenCA only includes the CA certificate which issued the enrolled certificate. Nevertheless it is possible to include as many certificate as you want.

Here are the steps to include other certificates into the PKCS#12 file - it is a typical Open Source solution:

1. Go to `OPENCADIR/lib/cmds`
2. Edit `send_cert_key_pkcs12`. There is a line

   ```
   my $cacert = getRequired ('....
   ```

   This line defines a file which includes all (CA) certificates which will be included into the PKCS#12 file. Usually we only include our CA certificate. Now you have to setup an individual for your chain.

   ```
   my $cacert = "/my/openca/dir/var/crypto/cacerts/blaine.pem";
   ```

3. The next step is to create an individual file for the chain. Now you have to create the file `blaine.pem`. This file has to include all needed CA certs in PEM format. Please remember to include a begin and end line before and after every CA certificate like for every normal PEM-formatted CA certificate.

4.18. Unknown login type

If you get an errormessage which indicates that there was an unknown login configured then please see Section 3.6, “Conflicting Modules”.

5. Access Control problems

5.1. Always get a login screen - again and again

You try to login with correct login and password. The result are one or two frames with the login screen again.

If your credentials are really correct then there is in the most cases a problem with the Perl module CGI::Session. Old versions of this module have problems with the flushing of data- especially the automatic flushing of data. OpenCA tries to handle this bug but the best solution is to remove the original module from your computer and to install a new one. You can install a new version from CPAN or you install OpenCA again. If OpenCA cannot find this module then it installs an actual version by itself.

5.2. Error 6251023: Aborting connection - you are using a wrong channel

This error message happens if you are using an apache without mod_ssl but you specified `mod_ssl` in your access control configuration.

5.3. Error 6251026: Aborting connection - you are using a wrong security protocol

This happens if the configuration requires `ssl` but you are using http to access the interface.

5.4. Error 6251029: Aborting connection - you are using the wrong computer

The access to the interface is only allowed for some computers with special IP addresses and you have not a computer with one of these special addresses.

5.5. Error 6251033: Aborting connection - you are using a wrong asymmetric cipher

There is a problem with the algorithms which mod_ssl and your browser are using. The configuration tries to enforce some special algorithms for security reasons. Please contact your administrators for more informations.

5.6. Error 6251036: Aborting connection - you are using a too short asymmetric keylength

The keylength of your certificate is too short if you are using a certificate. If you don't use a certificate then the Apache itself or your browser only support some too short asymmetric keys. Please contact your local administrators to find out more details.

5.7. Error 6251039: Aborting connection - you are using a wrong symmetric cipher

There is a problem with the algorithms which mod_ssl and your browser are using. The configuration tries to enforce some special algorithms for security reasons. Please contact your administrators for more informations.

5.8. Error 6251043: Aborting connection - you are using a too short symmetric keylength

If your browser startis a connection to webserver via https then the two software components try to negotiate about the cryptoalgorithms which they support. Usually they choose the strongest algorithm and longest keylength. Some export restricted browsers only support too short keylength. This is not allowed in security sensitive areas. Please ask your administrators for the exact specifications of your installation.

6. Dataexchange

6.1. I try to export something but I get error 512 “permission denied” for /dev/fd0

This is the most common problem with the export of data. The most modern Unix systems change the owner of local drives (if you can change the medias) to the UID and GID of the user who is logged in on the console. If you want to export something to the floppy then there are three choices:

1. The simplest solution is to change the permissions itself and to allow every other user to write on the disk.

```
chmod a+w /dev/fd0
```

2. If you run no X server or better no xdm (including gdm and kdm etc.) then it makes sense to simply change the group and perhaps the owner of the device. This operation makes no sense if you run xdm because there is a PAM module for device settings and this PAM module is usually included into /etc/pam.d/xdm.

```
chown wwwuser:wwwowner /dev/fd0
chmod u+w /dev/fd0
```

3. If you run OpenCA on a production system and you run an xdm on this system then you should use this way of changing the permissions but be prepared this is not very easy and you must test the changes very carefully.

   First you have to edit /etc/logindevperm. Please comment out the line which defines the settings for /dev/fd0. This will avoid the PAM module devperm from changing the owner (UID) and group (GID) of the device. Usually the PAM module is used by xdm (see /etc/pam.d/xdm) and other console based services.

   Now you can do the same like for the first or the second choice.

```
chown wwwuser:wwwowner /dev/fd0
chmod u+w /dev/fd0

or

chmod a+w /dev/fd0
```

   Now the change of the permissions is durable. The changes in the first and second choice are not durable if you use the PAM module devperm. Please don't wonder if a normal user cannot mount a floppy after these operations.

6.2. I try to import the CA certificate but it doesn't work.

Do you see the following during the import:

```
Test the archive ...
/bin/tar -tvf /dev/fd0
Importing archive ...
```

```
Load required variables ...
Changing to directory /usr/local/openca.0.9.1/openca/var/tmp/tmp_418 ...
Running the import command(s) ...
/bin/tar -xvf /dev/fd0 -C /usr/local/openca.0.9.1/openca/var/tmp/tmp_418
Importing the RBAC-configuration ... Ok.

LDAP-support is activated

Automatic LDAP-update is activated

Importing _CA_CERTIFICATE ...
No objects are present.
Importing CA-Certificates into ldap ...
Cannot load CA-certificate
Make CA-Certificate available on the server ...OK.

Re-Building CA Chain ... Ok.

Clean up ...Ok.
```

The important thing is the line:

```
Importing _CA_CERTIFICATE ...
```

If you see this line then your configuration for the dataexchange is wrong. This can happen if you installed the online components with `--with-hierarchy-level=`*`ca`*. A CA doesn't import of course a CA certificate. You can check the option `DOWNLOAD_CA_CERTIFICATE_STATES` in your configuration files. It should contain at minimum *`VALID`*.

6.3. I crashed the database of the online server and now I want to import all data again. How can I do it?

Go to `OPENCADIR/var/log/enroll` and `OPENCADIR/var/log/download`. There you have to remove all data from files which contain the module ID of the node interface on the online server which crashed. If you run the import/export commands next time then all objects will be exported again. It is the same technology like for the creation of a new node interface.

6.4. I try to export the requests to the CA but it doesn't work

Usually the log from the export shows no requests or looks like this:

Example C.12. Failed request upload

```
Exporting _REQUEST ...

FAILURE: 288 (spkac).

FILE: /srv/ra//OpenCA/var/tmp/tmp_1517/_REQUEST//288.spkac

FAILURE: 544 (spkac).

FILE: /srv/ra//OpenCA/var/tmp/tmp_1517/_REQUEST//544.spkac

Exporting archive ...
```

The symptom is “Exporting _REQUEST”. This shows that there are no status specified for the export of requests. New version of OpenCA simply ignore the requests and show nothing.

You can manually change the status of the requests which should be uploaded or you install again. In the most cases there was a wrong paramter for configure. Please check the value of `--with-hierarchy-level` very carefully. If you install a RA and you want to export approved requests to the CA then the hierarchy level must be *ra*.

7. LDAP

7.1. Errormessage: Connection refused.

This occurs if OpenCA cannot make a connection to the LDAP directory. Make sure that the ldap server is running and is listening on the correct port. Make sure that the settings in `ldap.xml` match your ldap server settings.

7.2. Errormessage: Bind failed. Errorcode 49.

A connection has been made to the ldap server, but the credentials to log into the server as admin are wrong. The bind operation is performed after the connection is established. Check the `login` (the LDAP administrator's DN) and `passwd` (the password of the ldap administrator).

7.3. The resultcode of the nodeinsertion was 65.

This sometimes means that OpenCA could not insert the appropriate entry for a certificate (the exact definition is `LDAP_OBJECT_CLASS_VIOLATION`). Check that you have the directory started with the appropriate schemas (core, cosine, inetorperson and openca). They are usually specified in `slapd.conf`.

7.4. How can I get more debugging messages from OpenCA's LDAP code?

You can get more debugging informations by turning on debugging in `OPENCADIR/etc/ldap.xml` (i.e. `<debug>1</debug>`). The most functions support this paramter.

7.5. How can I get more debugging messages from OpenLDAP?

The logging messages of OpenLDAP are sent to syslogd. OpenLDAP uses the facility local4. You can find the files which contain the logs in `/etc/syslog.conf`. Simply search for the files which will be used by local4.

If you need more informations than be in the log files from syslogd then you have to tune the configuration of OpenLDAP. Usually there is a file `/etc/openldap/slapd.conf` which contain the configuration. The logging information will be configured with the option `loglevel`. This is a bitmap with eleven bits today. A loglevel of 63 mean that the bits one to five are set. A good choice is 63 for a first debugging session. You can read the details in **man slapd.conf**.

8. Internationalization

8.1. How can I fix a misspelling for a language?

8.2. How can I add a new langiage?

8.3. The compilation/make fails on the Perl module gettext

Sometimes there are problems with the compilation of the Perl module gettext. The most common problem is a loader problem where ld reports a non existing library intl (`-lintl`). The output look like follows:

Failing gettext compilation

```
make[4]: Entering directory `/root/openca-0.9.1.3/src/modules/gettext-1.01'
cp gettext.pm blib/lib/Locale/gettext.pm
/usr/bin/perl5.8.1 /usr/lib/perl5/5.8.1/ExtUtils/xsubpp  -typemap
/usr/lib/perl5/5.8.1/ExtUtils/typemap  gettext.xs > gettext.xsc && mv
gettext.xsc gettext.c
Please specify prototyping behavior for gettext.xs (see perlxs manual)
gcc -c   -D_REENTRANT -D_GNU_SOURCE -DTHREADS_HAVE_PIDS
-fno-strict-aliasing -I/usr/local/include -D_LARGEFILE_SOURCE
-D_FILE_OFFSET_BITS=64 -I/usr/include/gdbm -O2 -fomit-frame-pointer -pipe
-march=i586 -mcpu=pentiumpro   -DVERSION=\"1.01\" -DXS_VERSION=\"1.01\"
-fPIC "-I/usr/lib/perl5/5.8.1/i386-linux-thread-multi/CORE"   gettext.c
gettext.c: In function `XS_Locale__gettext_gettext':
gettext.c:67: warning: assignment makes pointer from integer without a cast
gettext.c: In function `XS_Locale__gettext_dcgettext':
gettext.c:86: warning: assignment makes pointer from integer without a cast
gettext.c: In function `XS_Locale__gettext_dgettext':
gettext.c:104: warning: assignment makes pointer from integer without a cast
gettext.c: In function `XS_Locale__gettext_textdomain':
gettext.c:121: warning: assignment makes pointer from integer without a cast
gettext.c: In function `XS_Locale__gettext_bindtextdomain':
gettext.c:139: warning: assignment makes pointer from integer without a cast
Running Mkbootstrap for Locale::gettext ()
chmod 644 gettext.bs
rm -f blib/arch/auto/Locale/gettext/gettext.so
LD_RUN_PATH="" gcc  -shared -L/usr/local/lib gettext.o  -o
blib/arch/auto/Locale/gettext/gettext.so   -lintl
/usr/bin/ld: cannot find -lintl
collect2: ld returned 1 exit status
make[4]: *** [blib/arch/auto/Locale/gettext/gettext.so] Erreur 1
make[4]: Leaving directory `/root/openca-0.9.1.3/src/modules/gettext-1.01'
make[3]: *** [gettext-1.01] Erreur 2
make[3]: Leaving directory `/root/openca-0.9.1.3/src/modules'
make[2]: *** [modules] Erreur 2
make[2]: Leaving directory `/root/openca-0.9.1.3/src/modules'
make[1]: *** [modules] Erreur 2
make[1]: Leaving directory `/root/openca-0.9.1.3/src'
make: *** [src] Erreur 2
```

The Perl module is only an interface to a C library from gettext. You must install the package gettext and the problem should be solved.

Bibliography

OpenSSL's webpage [http://www.openssl.org]. OpenSSL project.

OpenCA's webpage [http://www.openca.org]. OpenCA project.

PKIX [http://www.ietf.org/html.charters/pkix-charter.html]. IETF.

RFC 2246 [http://www.ietf.org/rfc/rfc2246.txt] The TLS Protocol Version 1.0. T. Dierks and C. Allen. IETF. January 1999.

RFC 2253 [http://www.ietf.org/rfc/rfc2253.txt] LDAP - UTF-8 String Representation of Distinguished Names. IETF. December 1997.

RFC 2311 [http://www.ietf.org/rfc/rfc2311.txt] S/MIME: Version 2 Message Specification. S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka. IETF. March 1998.

RFC 2312 [http://www.ietf.org/rfc/rfc2312.txt] S/MIME: S/MIME Version 2 Certificate Handling. S. Dusse, P. Hoffman, B. Ramsdell, and J. Weinstein. IETF. March 1998.

RFC 2315 [http://www.ietf.org/rfc/rfc2315.txt] PKCS #7: Cryptographic Message Syntax Version 1.5. B. Kalinski. IETF. March 1998.

RFC 2595 [http://www.ietf.org/rfc/rfc2595.txt] Using TLS with IMAP, POP3 and ACAP. C. Newman. IETF. Jun 1999.

RFC 2828 [http://www.ietf.org/rfc/rfc2828.txt] Internet Security Glossary. R. Shirey. IETF. May 2000.

RFC 2818 [http://www.ietf.org/rfc/rfc2818.txt] HTTP Over TLS. E. Rescorla. IETF. May 2000.

RFC 2898 [http://www.ietf.org/rfc/rfc2898.txt] PKCS #5: Password-Based Cryptography Specification Version 2.0. B. Kaliski. IETF. September 2000.

RFC 2985 [http://www.ietf.org/rfc/rfc2985.txt] PKCS #9: Selected Object Classes and Attribute Types Version 2.0. M. Nystrom and B. Kaliski. IETF. November 2000.

RFC 2986 [http://www.ietf.org/rfc/rfc2986.txt] PKCS #10: Certification Request Syntax Specification Version 1.7. M. Nystrom and B. Kaliski. IETF. November 2000.

RFC 3207 [http://www.ietf.org/rfc/rfc3207.txt] SMTP Service Extension for Secure SMTP over Transport Layer Security. P. Hoffman. IETF. February 2002.

RFC 3279 [http://www.ietf.org/rfc/rfc3279.txt] Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. L. Bassham, W. Polk, and R. Housley. IETF. April 2002.

[RFC3280] *RFC 3280 [http://www.ietf.org/rfc/rfc3280.txt] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. R. Housley, W. Polk, W. Ford, and D. Solo. IETF. April 2002.

RFC 3447 [http://www.ietf.org/rfc/rfc3447.txt] Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. J. Jonsson and B. Kaliski. IETF. February 2003.

OpenLDAP's webpage [http://www.openldap.org]. OpenLDAP.

PKI - Implementing and Managing E-Security. Andrew Nash, William Duane, Celia Joseph, and Derek Brink. Coriolis Group. 2001.

LDAP, Programming Directory-Enabled Applications with Lightweight Directory Access Protocol. Timothy A. Howes and Marc C. Smith. Macmillan Technology Series, Macmillan Technical Publishing. 1997.

Understanding and Deploying LDAP Directory Services. Timothy A. Howes, Marc C. Smith, and Gordon S. Good. Macmillan Network Architecture and Development Series, Macmillan Technical Publishing. 1999.

Public Key Infrastructure for KMU's (Project report, German only). Stefan Dietiker and Chris Berger. Zuericher Hochschule Winterthur. Jule 2003.

Aufbau und Betrieb einer Zertifizierungsinstanz, DFN-PCA Handbuch (German only). I. Camphausen, St. Kelm, B. Liedtke, and L. Weber. Deutsches Forschungsnetz - DFN (Germany`s National Research and Education Network). May 2000.

Homepage of the SET standard [http://www.setco.org]. SET Secure Electronic Transaction LLC.

DocBook Stylesheet Development on SourceForge [http://docbook.sourceforge.net/]. DocBook.

DocBook XSL: The Complete Guide [http://www.sagehill.net/docbookxsl/]. Bob Stayton. Sagehill Enterprises. September 2003.

DocBook: The Definitive Guide [http://www.docbook.org/tdg/en/html/]. Norman Walsh, Leonard Muellner, and With contributions from Bob Stayton. O'Reilly & Associates. 31st December 2003.

The DocBook Wiki [http://docbook.org/wiki/moin.cgi/]. DocBook.

Glossary

CA	Certification Authority
CRL	Certificate Revocation List
CRR	Certificate Revocation Request
CSR	Certificate Signing Request
DMZ	DeMilitarized Zone is an area which is isolated from the inner and outer network of a firewall system. It is used to place servers in a protected area which has no direct access to the inner and outer network but can offer this service to people or systems in both areas. This is a very short description. Please consult specialized books or even better humans if you have absolute no idea how firewall systems work. This is really security relevant.
DN	Distinguished Name
HTTPS	is nothing else than HTTP trough a SSL or TLS tunnel. It protects the communication between a http server and a browser. It was the first application for SSL and should be today the world's most widely used SSL application.
IMAPS	is nothing else than IMAP trough a SSL or TLS tunnel. It protects the communication between a mail server and a mail user agent if the user reads and manage it's mail.
LDAP	Lightweight Directory Access Protocol
LOA	Level Of Assurance defines the quality of the identification of the certificate owner. Sometimes it is usefule to know how the owner of certificate was identified or would you send money because of signed mail if the owner was identified via email?
MTA	Mail Transfer Agent - a tool to send mail to other users or mail servers, e.g. Mozilla, Outlook (Express). Sendmail can be a MTA too if it acts as client.
MUA	Mail User Agent - e.g. Mozilla, Outlook (Express). This is the tool which a user uses to read and handle it's mail.
Node	This is the management interface for an OpenCA installation on one machine.
PKCS	Public Key Cryptography Standards are developed by RSA Security. They are widely accepted in the PKI area.
PKCS#10	defines the ASN.1 structure of certificate signing request
POPS	is nothing else than POP trough a SSL or TLS tunnel. It protects the communication between a mail server and a mail user agent if the user reads and manage it's mail.
RA	Registration Authority
SCEP	Simple Certificate Enrollment Protocol was developed by Cisco and is used to handle the communication between a PKI and networkcomponents like router, switches and other (perhaps software) VPN components.

S/MIME	Secure MIME is a standard which defines how secured emails must be formatted. Please check the listed RFCs to find references to more detailed descriptions.
SMTP	Simple Mail Transfer Protocol is used by mailservers like sendmail to exchange the mails. The protocol is used for mailtransfer from simple MTAs like Mozilla to servers like sendmail and for transfers from server to server.
SMTP over TLS	is nothing else than SMTP trough a TLS tunnel. It protects the communication between a mail server and a mail user agent if the user reads and manage it's mail.
SPKAC	Signed Public Key And Challenge is a standard for CSRs from Netscape.
SSL	Secure Socket Layer is a tranport layer security protocol. It was one of the first certificate based security protocols for tunneling. Netscape developed this protocol for it's webbrowser Navigator. TLS is the standardized follower of this protocol. Today the versions 2 and 3 are still widely used and supported.
subject	The subject of a certificate or of a request is the name of the certificate or request. The subject is a distinguished name and looks like “cn=Jon Doe, ou=Sales, o=startup, c=us”.
Symlink	A symlink is nothing else than a symbolic link. Such links will be created by OpenCA usually with **ln -s**. We always try to avoid the shortcut but sometimes we are simply to fast ;-)
TLS	Transport Layer Security is a tranport layer security protocol. It is the standardized successor of SSL. All modern browsers use this protocol but it can be used to tunnel every other TCP based service.