

OpenCA Guide for Versions 0.9.2+

OpenCA Guide for Versions 0.9.2+

Copyright © 2002, 2003, 2004, 2005 OpenCA Group TM

Table of Contents

Introduction	xv
I. Design Guide	17
Preface	xix
1. General Design	20
1. Basic Hierarchy	20
2. Interfaces	21
2.1. Node	22
2.2. CA	22
2.3. RA	22
2.4. LDAP	22
2.5. Pub	22
3. Configuration	23
4. Database	23
5. Interface	23
6. Life cycle of the objects	23
7. Sub-Ca	24
7.1. Example 1	24
7.2. Example 2	24
2. Recommendations	25
1. Hardware Issues	25
1.1. Time	25
1.2. Failing disks	25
1.3. Hardware monitoring	25
2. Physical Security	26
2.1. Safes and Data organization	26
2.2. Buildings	26
3. Network Issues	26
4. Certificate Issues	27
4.1. CDPs	27
4.2. Application specific problems	28
5. Organizational Aspects	28
5.1. Dual Access Control	28
5.2. Privacy vs. Security	28
5.3. Enforcement of Access Control	28
5.4. Privacy Officer Integration	28
5.5. Enterprise Integration	29
5.6. Parallel use of several end user PKIs	29
II. Installation and Configuration Guide	30
Preface	xxxiii
3. Installation	34
1. Preparations	34
1.1. Software	34
1.2. Hardware	35
2. Configure	35
2.1. Host System Configuration	35
2.2. Host System Configuration (of the upcoming OpenCA 1.0)	36
2.3. Filesystem paths	36
2.4. Webserver specific stuff	37
2.5. Email	38
2.6. Compiling features	38
3. Installation	38
4. config.xml (for RPMs and DEBs too)	39
4.1. Configuration sections of config.xml	39

4.2. How to setup two management interfaces on one server?	42
4. Configuration	43
1. Access Control	43
1.1. Channel verification	44
1.2. Login	44
1.3. Session management	47
1.4. ACLs	48
2. Token and keyconfiguration	50
2.1. OpenSSL	52
2.2. Empty	52
2.3. LunaCA3	52
2.4. nCipher	53
2.5. OpenSC	59
3. OpenSSL	59
3.1. Certificate Extensions	60
3.2. Profiles	61
4. CSRs	63
4.1. Additional Attributes	63
4.2. PKCS#10 Requests	64
4.3. Basic CSR	64
4.4. SCEP	66
5. Subject	66
5.1. Common stuff	66
5.2. dc style	67
6. Subject Alternative Name	68
7. LDAP	69
7.1. Configuration of the Directory	69
7.2. Configuration of the online components	69
7.3. Writing Certificates to the Directory	71
7.4. Adding an attribute to the LDAP schema	71
8. SCEP	72
8.1. OPENCADIR/etc/servers/scep.conf	72
8.2. OPENCADIR/etc/config.xml	74
9. Dataexchange	74
9.1. Configuration	74
9.2. Adding a new node	77
10. Databases	77
10.1. PostgreSQL	77
10.2. MySQL	79
10.3. Oracle	80
10.4. DBM Files	84
10.5. SQLite	85
11. Email	85
11.1. Sendmail with basic SMTP authentication	85
12. i18n	87
12.1. Debian 3.1 Sarge	87
III. User Guide	88
Preface	xcv
5. Features	92
1. 0.10	92
2. 0.9.2	92
6. Interface Descriptions	93
1. Public PKI Server	93
1.1. General	93
1.2. CA Infos	93
1.3. User	94
1.4. Certificates	96
1.5. Requests	97

1.6. Language	97
2. Registration Authority	97
2.1. General	98
2.2. Active CSRs	98
2.3. Active CRRs	99
2.4. Information	99
2.5. Utilities	101
3. Registration Authority Node	101
3.1. General	101
3.2. Administration	102
3.3. Utilites	104
3.4. Logs	104
4. LDAP Interface	105
4.1. Update LDAP	105
4.2. View CA-Certificates	105
4.3. View Certificates	106
4.4. View CRLs	107
7. Functionality Descriptions	109
1. CA Initialization	109
1.1. Phase I: Initialize the Certification Authority	109
1.2. Phase II and III: Create the initial administrator and RA certificate	112
2. Node Initialization	113
3. CSR Handling - a request HOWTO	114
3.1. Ways to request a certificate	114
3.2. Edit a certificate signing requests	117
3.3. Approve certificate signing requests	117
3.4. Issue a certificate from a certificate signing request	117
3.5. Certificate enrollment	118
3.6. Delete certificate signing requests	118
4. Certificate Handling	118
4.1. Find a certificate	118
4.2. Download	118
4.3. Start revocation	119
4.4. Write an email to the owner	119
4.5. Informational messages and their meaning	119
5. SCEP	119
5.1. SSCEP	119
5.2. NetScreen ScreenOS	121
5.3. F-Secure VPN+	121
5.4. Cisco PIX	122
8. Client Support	123
1. Introduction	123
2. Mozilla	123
2.1. General	124
2.2. Mozilla	124
2.3. Netscape 4	125
2.4. Opera	125
3. Microsoft	125
3.1. Domaincontroller	125
3.2. Smartcard Logon	128
3.3. Keystore	129
3.4. Internet Explorer	129
3.5. Outlook	130
3.6. Outlook Express	130
IV. Technology Guide	131
Preface	cxxxiv
9. Introduction	135
1. Slotchnology	135

10. XML	137
11. Cryptolayer	138
12. Accesscontrol	140
13. Logging	145
14. Webinterfaces	147
1. Interfacebuilding	147
1.1. Technology overview	147
1.2. Customization capabilities	149
2. CSS	150
3. Configuration after installation	150
15. Hierarchy	151
1. Nodemangement	151
2. Dataexchange	151
16. LDAP	152
1. LDAP schema specification	152
1.1. Used objectclasses	152
1.2. Supported attributes	152
1.3. Common definitions for distinguished names	153
1.4. Special definitions for user certificates	154
2. Sourcecodeorganization	154
2.1. Structure of the code	154
2.2. The relevant commands	155
2.3. export-import.lib	155
2.4. ldap-utils.lib	155
2.5. OpenCA::LDAP	155
17. Database	156
1. Tables	156
2. Sequences	157
3. Indexes	157
18. Batch System	158
1. Requirements	158
2. Design	158
3. Data Import	159
4. Database background	161
5. Change the workflow	162
6. Default workflow	163
7. What about the different crypto tokens?	164
8. Performance	164
8.1. PIII 850MHz, 256 MB RAM	164
19. Packaging	165
1. Common Notices	165
1.1. Required Perl modules	165
2. RPM-based system	165
2.1. RedHat/Feodora	165
2.2. SuSE	165
3. Debian	167
4. BSD	167
20. Software Design (legacy from design guide)	168
1. Database(s)	168
2. Interface construction	168
3. openca.cgi	168
4. libraries	168
5. modules	168
6. commands	168
7. Dataexchange and Node management	168
A. History	169
1. PKI Scenario before OpenCA	169
2. PKI and eGovernment	170

3. Internet Standards	170
4. The Project's Purposes	170
5. The Project's Achievements	171
6. The OpenCA Project	171
6.1. The project start	171
6.2. Offering Help to Other Projects: OpenSSL	172
6.3. CVS and Mailing Lists	172
6.4. The Open Source Choice	172
6.5. Migrating to SourceForge	173
B. References	174
1. Universities	174
C. Internationalization - i18n	175
1. de_DE	175
2. it_IT	175
3. ja_JP	175
4. pl_PL	175
5. sl_SI	175
D. Authors and Contributors	176
1. Martin Bartosch	176
2. Michael Bell	176
3. Chris Covell	176
4. Massimiliano Pala	176
5. Ulrich Bathels	176
6. Ashutosh Jaiswal	176
7. FAQ	176
E. FAQ	177
1. General PKI Issues	177
1.1. What is a certificate?	177
1.2. Which informations does a certificate contain?	177
1.3. What is a request?	177
1.4. Which information does a <i>CSR</i> contain?	177
1.5. What is a <i>CA</i> ?	178
1.6. Why should I not place the <i>CA</i> on the same machine like the <i>RA</i> ?	178
1.7. What is an extensions?	178
1.8. I use Windows 2000 and Internet Explorer 6 SP1 and it don't show any CSPs.	178
1.9. How can I setup a sub <i>CA</i> ?	178
2. General OpenCA Issues	178
2.1. Does it be possible to revoke a certificate without any user interaction?	179
2.2. I try to add a role and get the message "The role XYZ exists already!"	179
2.3. All cryptographic operations fail.	179
2.4. Apache's error_log reports a nonexistent option <code>-subj</code> of openssl req	179
2.5. Apache's error_log contains a message from IBM DB2 that the environment is not setted	179
2.6. What do the new features of 0.9.2 be?	179
2.7. I try to approve and sign a request with Mozilla and it fails.	180
2.8. I try to approve and sign a request with Konqueror (KDE) and it fails.	180
2.9. How is the format of the disc to import the <i>CA</i> certificate from the root <i>CA</i> ?	180
2.10. OpenSSL reports entry 1: invalid expiry date	180
2.11. Outlook cannot encrypt mail with imported certificate	180
2.12. My Outlook freezes after I received a signed email	180
2.13. General Error 6751 during certificate issuing	181
2.14. What does I have to do if I create a new release?	181
2.15. How can I configure Mozilla for OCSP?	182
2.16. Error 7211021: Cannot create request!	182
3. Installation Issues	183
3.1. FreeBSD, OpenBSD and OpenCA	183
3.2. Solaris and OpenCA	184
3.3. What is a hierarchy level?	184
3.4. Undefined subroutine <code>&main::xyz</code>	184

3.5. Symbolic link installaton failed	185
3.6. After the installation all common parts are missing	185
3.7. Conflicting Modules	185
3.8. The xml path to the access control is missing	186
3.9. Unknown Login Type	186
3.10. Type Mismatch during request generation with Internet Explorer	186
3.11. openca(_rc) start failed	187
3.12. Missing modules	187
3.13. Trouble with databases, database drivers and Perl DBI	189
4. Configuration Issues	189
4.1. How can I configure my httpd.conf for virtual hosts?	189
4.2. How can I configure virtual hosts with ./configure?	190
4.3. I have some users which should not be published in LDAP. Does it be possible with OpenCA?	190
4.4. Does it be possible to authenticate users by their certificates at the apache before they will be authenticated by OpenCA itself?	190
4.5. I want to update my 0.9.2 installation. Is this dangerous?	191
4.6. I want update to 0.9.2. How can I update my sql database?	191
4.7. If I run openca-ocspd then I obtain a segmentation fault.	192
4.8. I installed a second public interface, run configure_etc.sh and now are all the paths in the other public interface wrong.	193
4.9. I issue a certificate for a mailserver but sendmail doesn't work and reports an errormessage which includes "reason=unsupported certificate purpose"	193
4.10. My (Microsoft) client hangs after it tries to start a secured connection	193
4.11. Outlook freezes when receiving a signed Mail but worked already fine for some days ...	194
4.12. During the request generation OpenCA fails and reports a too short textfield	194
4.13. Can I place my organization's logo on the web interface?	194
4.14. Cannot create new OpenCA tokenobject	194
4.15. How can I use a Luna token with OpenCA 0.9.1	195
4.16. How can I include a complete certificate chain into a PKCS#12 file?	195
4.17. Unknown login type	196
4.18. Cannot initialize cryptoshell but OpenSSL path is correct	196
4.19. Emailaddress in subjectAltName but not in CA subject	196
4.20. Missing environment variables from SSL	197
4.21. Problems with the country name during PKCS#10 requests	197
5. Access Control problems	198
5.1. Always get a login screen - again and again	198
5.2. Error 6251023: Aborting connection - you are using a wrong channel	198
5.3. Error 6251026: Aborting connection - you are using a wrong security protocol	198
5.4. Error 6251029: Aborting connection - you are using the wrong computer	198
5.5. Error 6251033: Aborting connection - you are using a wrong asymmetric cipher	198
5.6. Error 6251036: Aborting connection - you are using a too short asymmetric keylength ...	199
5.7. Error 6251039: Aborting connection - you are using a wrong symmetric cipher	199
5.8. Error 6251043: Aborting connection - you are using a too short symmetric keylength	199
6. Dataexchange	199
6.1. I try to export something but I get error 512 "permission denied" for /dev/fd0	199
6.2. I try to import the CA certificate but it doesn't work.	200
6.3. I crashed the database of the online server and now I want to import all data again. How can I do it?	201
6.4. I try to export the requests to the CA but it doesn't work	201
7. LDAP	201
7.1. Errormessage: Connection refused.	201
7.2. Errormessage: Bind failed. Errorcode 49.	201
7.3. The resultcode of the nodeinsertion was 65.	202
7.4. How can I get more debugging messages from OpenCA's LDAP code?	202
7.5. How can I get more debugging messages from OpenLDAP?	202
8. Internationalization	202
8.1. How can I fix a misspelling for a language?	202

8.2. How can I add a new language?	202
8.3. The compilation/make fails on the Perl module gettext	202
8.4. MySQL and SET NAMES errormessages	203
Bibliography	204
Glossary	206
F. Strategy	208
1. The Strategy Behind OpenCA Development	208
1.1. Scalability	208
1.2. Command Line API to CA and RA Functions	208
1.3. Automation functions	208
1.4. On-line CA model option	208
1.5. High Risk Environment Mode	208
1.6. Audit logging	208
1.7. Script/environment validation	208
1.8. Automated CA rollover	209
1.9. Function to process signing and encryption keys in one go	211
1.10. Secure storage and recovery of encryption keys	211
1.11. Web based OpenCA configuration and management	211
1.12. Improved key lifecycle management	211
1.13. Authentication via a third party	211
1.14. Improved debugging support	211
1.15. Improved error handling	211
1.16. Accreditation	212

List of Figures

1.1. Database oriented view	20
1.2. Logical data view	20
1.3. Complete technical overview	21
1.4. Life cycle of objects	23
4.1. Passes of the accesscontrol	43
4.2. Passphrase based login	45
4.3. Tokenconcept	50
7.1. Phases of the CA initialization	109
7.2. Phase I of the CA initialization	110
7.3. Phase II of the CA initialization	112
7.4. Phase III of the CA initialization	112
8.1. Request a certificate	123
11.1. Example cryptolayer with tokens	138
12.1. Passes of the accesscontrol	140
12.2. Channel verification	141
12.3. Identification of the user	142
12.4. Access control list	143
16.1. LDAP source schema	154

List of Tables

3.1. External Perl modules	34
3.2. Supported parameters for host configuration	35
4.1. Additional attributes configuration	63
4.2. Generic basic CSR configuration	65
4.3. Common stuff configuration	67
16.1. Schema usage	153
16.2. Schema usage for user certificates	154
18.1. Default OpenCA workflow	163
18.2. 1000 User test	164
E.1. Texttypes for different databases	192

List of Examples

3.1. Module ID calculation	41
4.1. channel configuration	44
4.2. Login and Passphrase configuration	46
4.3. External program authentication configuration	46
4.4. Authentication with certificates	47
4.5. Session configuration	48
4.6. Basic ACL configuration	48
4.7. Permission for serverInfo	49
4.8. Allow all	49
4.9. Configuration of HSM Login/logout in menu.xml	53
4.10. Configuration of token.xml for nCipher	56
4.11. OpenSSL configuration - Authority Key Identifier	60
4.12. Minimal SSL client extensions	62
4.13. Minimal SSL server extensions	62
4.14. Minimal SMTP extensions for a single certificate	62
4.15. Additional attributes configuration	63
4.16. PKCS#10 configuration	64
4.17. Basic CSR configuration	64
4.18. Configuration example for a XML file based HTML-select	66
4.19. suffix in ldap.xml	70
4.20. excluded roles in ldap.xml	71
4.21. Schema extension for RDN uid_special	72
4.22. Download configuration	74
4.23. Export configuration	75
4.24. Local export configuration	75
4.25. OpenCA rc script that sources Oracle environment	80
4.26.	82
4.27. /etc/mail/sendmail.mc	85
7.1. SSCEP configuration	120
8.1. OpenSSL 0.9.7 key usage and extended key usage for DCs	126
8.2. OpenSSL 0.9.7 subjectAltName for DCs	127
8.3. OpenSSL 0.9.7 certificate template name for DCs	127
8.4. OpenSSL 0.9.8 subject alternative name section for DCs	128
8.5. extendedKeyUsage for clients	129
18.1. batch_new_user.txt	160
18.2. batch_new_process.txt	160
18.3. batch_process_data.txt	161
19.1. SuSE packaging	166
E.1. General error 6751 during certificate issuing	181
E.2. Bad passphrase error log during certificate issuing	181
E.3. Error 7211021: Cannot create request!	182
E.4. Full error message for missing functions	184
E.5. Already present symbolic link	185
E.6. Search for XML::Twig modules	186
E.7. Type Mismatch on Internet Explorer	187
E.8. Failed startup with wrong Net::Server version	187
E.9. failing XML parsing during configuration	187
E.10. failing XML parsing during configuration	188
E.11. empty Twig.pm files because of missing XML::Parser	188
E.12. Too old DBD::Pg or DBI trouble	189
E.13. virtual host configuration	189
E.14. ./configure and virtual hosts	190
E.15. Client authentication with mod_ssl	190

E.16. OCSP configuration for LDAP	192
E.17. OCSP configuration for http	193
E.18. emailaddress for subjectAltName in CA certs	197
E.19. Missing mod_ssl standard environment variables	197
E.20. SSL environment variable configuration for Apache	197
E.21.	198
E.22. Failed request upload	201

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
- Password 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	xix
1. General Design	20
1. Basic Hierarchy	20
2. Interfaces	21
2.1. Node	22
2.2. CA	22
2.3. RA	22
2.4. LDAP	22
2.5. Pub	22
3. Configuration	23
4. Database	23
5. Interface	23
6. Life cycle of the objects	23
7. Sub-Ca	24
7.1. Example 1	24
7.2. Example 2	24
2. Recommendations	25
1. Hardware Issues	25
1.1. Time	25
1.2. Failing disks	25
1.3. Hardware monitoring	25
2. Physical Security	26
2.1. Safes and Data organization	26
2.2. Buildings	26
3. Network Issues	26
4. Certificate Issues	27
4.1. CDPs	27
4.2. Application specific problems	28
5. Organizational Aspects	28
5.1. Dual Access Control	28
5.2. Privacy vs. Security	28
5.3. Enforcement of Access Control	28
5.4. Privacy Officer Integration	28
5.5. Enterprise Integration	29
5.6. Parallel use of several end user PKIs	29

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 be changed at every time the software changes. So please be lenient toward us :)

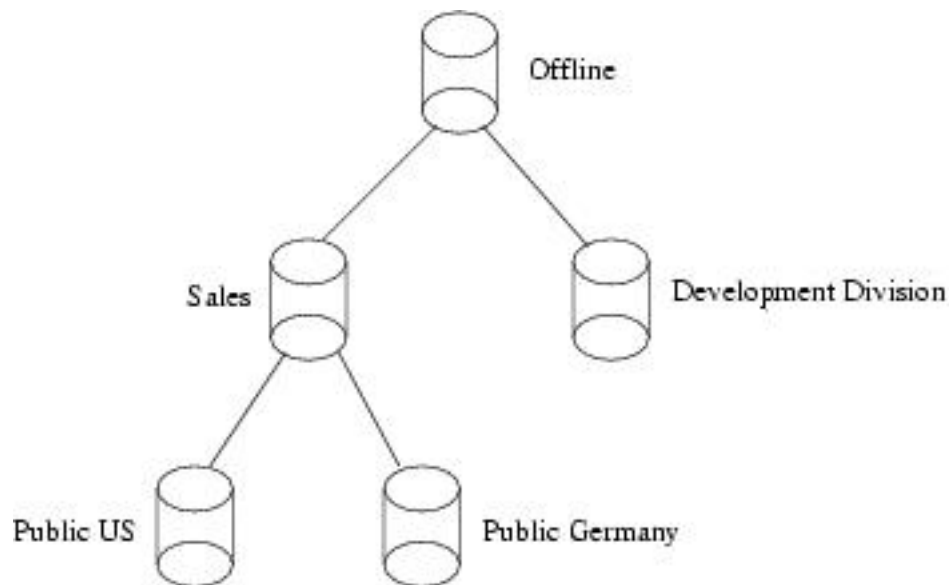
Chapter 1. General Design

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 OpenCA novices must also have a chance to understand the software.

1. Basic Hierarchy

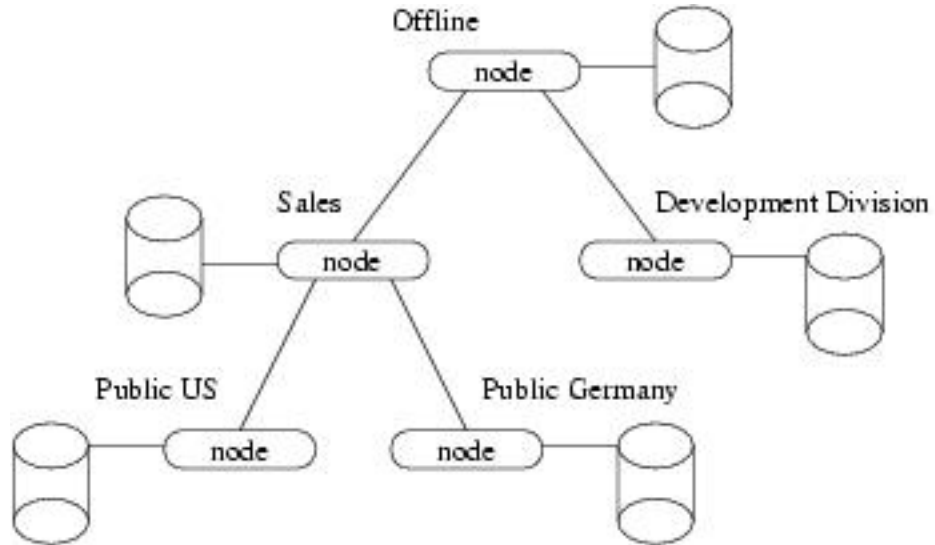
The basic idea of every X.509 PKI (Public Key Infrastructure) is a strong hierarchical organization. This results in a tree of databases if we try to create a distributed PKI architecture.

Figure 1.1. Database oriented view



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

Figure 1.2. Logical data view



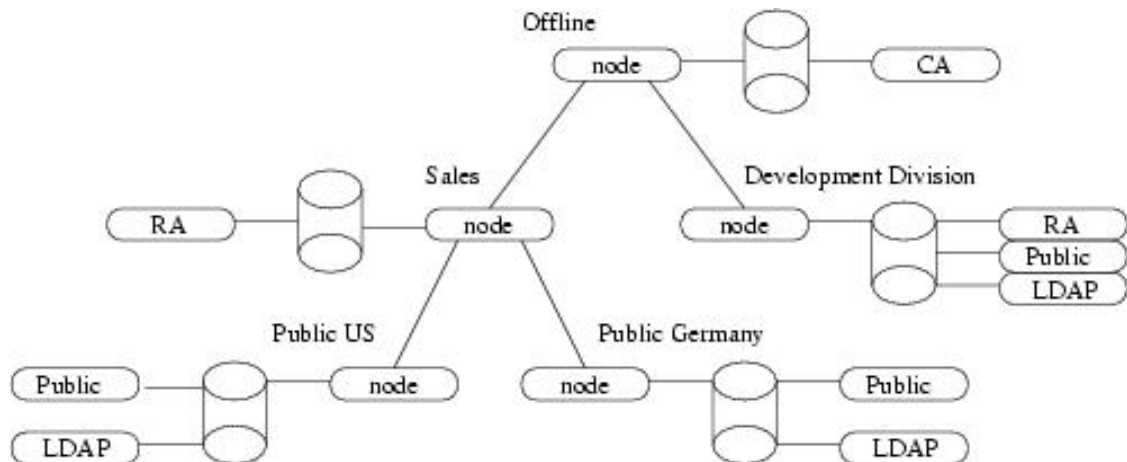
Normally every server in the infrastructure of the trustcenter has it's own database for security reasons. This hierarchy is the backbone of the trustcenter.

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 web-gateway? OpenCA supports all these software components via special web interfaces.

If you want to design a powerful trustcenter then you must have a concept about how you want to organize your work flow. You can see an example in the following figure.

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

The CA is the home of the batch processors too. OpenCA includes some powerful batch processors for creating certificates. These batch processors can be used for automatic certificate creation from various Enterprise Resource Planning (ERP) systems (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 smart cards, delete wrong requests and email users.

2.4. LDAP

The LDAP interface was implemented to separate the LDAP management completely from the rest of the software. This is necessary because there are many functions which are really specific for LDAP admins, with only a few users needing these features.

2.5. Pub

The Public interface includes 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 client independent requests and private keys (e.g. for KDE's konqueror or server administrators who 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 user certificates in browsers (Microsoft Internet Explorer and Netscape Communicator and Navigator 4.7x)

3. Configuration

4. Database

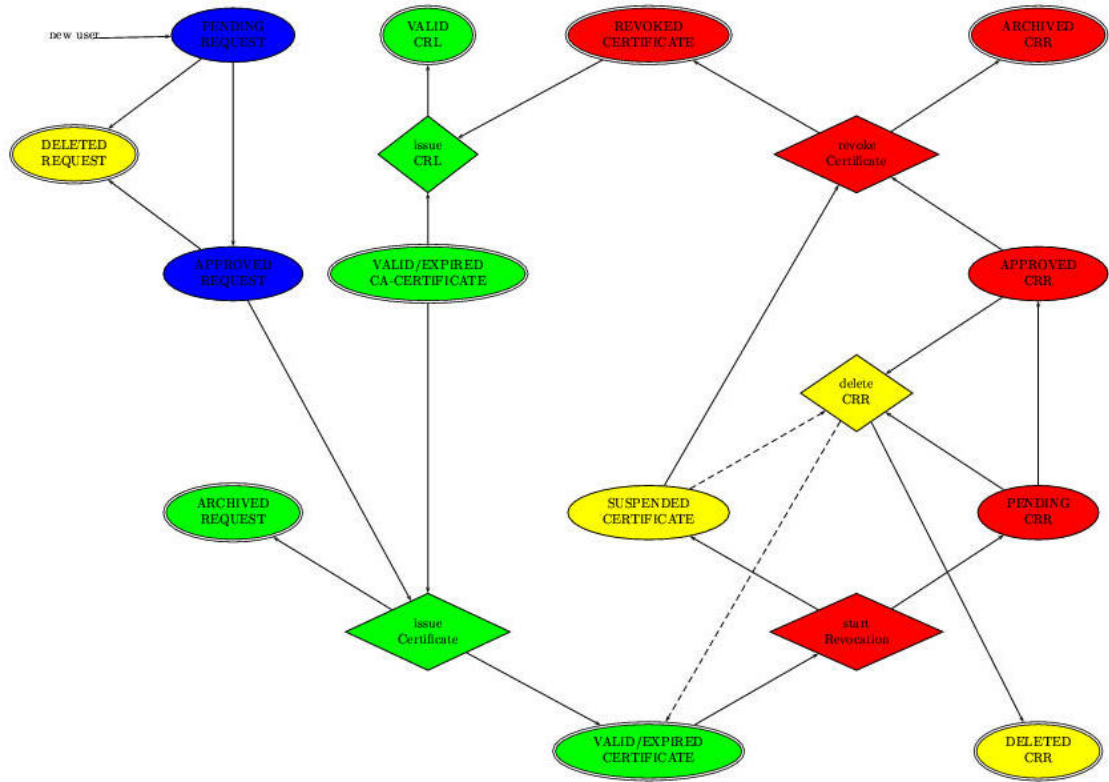
5. Interface

6. Life cycle of the objects

OpenCA has a very private key centric workflow idea. This means that all objects which deal with the same public/private key should be connected to each other. The reason is that the most dangerous event is a key compromise. If only a single certificate is wrong then we have no problem but if a key is compromised then all requests and certificates which are connected to this key are affected.

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

Figure 1.4. Life cycle 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 their production 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 time source. The most timeservers finally use a radio clock which receives a unsigned time signal from a radio station. This signal can be easily faked because it is very weak. So network time sources are really insecure.

After discussing the disadvantages of using online computers we can discuss offline technologies. Radio clocks are problematic and hence we need not discuss them twice. Also many buildings made from good ferroconcrete do not have problems with radio signals because they act as really nice Faraday cages! So what are the alternatives? First we need a trustworthy time source. Simply take a digital watch and compare it's time with several other clocks on the Internet, the video text of your TV, a radio clock, 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. So as you can see time is not trivial.

1.2. Failing disks

The most common hardware crashes involve cooler and disk failures. You should backup all your important data - especially 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 disk crashes. RAIDs are sometimes expensive but they helps a lot (ask your SAN admins :)).

1.3. Hardware monitoring

Usually you can make a visual observation if your laptop crashes. A crashed offline computer can be detected by visual monitoring too. However a crashed online component of a PKI is problem because important information is not available, such as newly issued certificates and CRLs. Such a situation also brings down your services like SCEP. If a public interface of the security infrastructure is down then you are bound to have a trust problem with your users in the future. It is also noteworthy that a simple ping is not enough to detect hardware failure. 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 Data organization

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 use the following method as an example: use one IT module safe and two data safes. An IT module safe is a safe with its own climatization 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. The setup 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 immediately start a roll out of a new infrastructure but there is no reason for panic. This arrangement also ensures that a loss of one safe doesn't stop your operation. You need two safes to survive and the loss 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 use more complex schemes to tolerate more than one broken safe. They should be able to tolerate at least 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 solid walls and a door with two locks. The room including the climatization system should be fire proof 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 should also have an alarm system and must be safe against electro magnetic pulse (EMP) and water flooding. This is only a short summary of ideas. Please ask some assurance specialists or architects for more details.

3. Network Issues

A PKI is only fully functional if all the services provided by it are fully operational. This include not only things like OCSP and SCEP but the public gateway as well. Many people think it is enough if the OCSP and one CDP keeps working but this is not sufficient. The first reason being that most applications don't understand OCSP. The second problem is that the last running CDP might only support LDAP, regardless of the fact that there are applications which only support HTTP. Even more problematic is a single running HTTPS CDP. The core mistake in the assumption is the meaning of fully operational. A PKI is not fully operational if only the CDPs still work. Nobody can download a new certificate or the certificate of an existing user in such a state. The PKI would still be secure but it would not be 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

fiber only helps if it is not in same pipe like the other one. Digging crews don't differentiate between the fibers if they accidentally cut a pipe! I know this situation really well :(If you have big distributed unit then it is recommended that at the minimum two of these units run the public interface. In this case you should also have independent interconnects.

4. Certificate Issues

This section consists of two big groups of problems - CDPs and application specific problems. Applications are also discussed in other sections of the OpenCA guide in case we have developed special configurations to solve the problem.

4.1. CDPs

CRL Distribution Points are a critical area. There were several PKIs in the earlier years which had 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 should not be a source for a CRL. It can be an OCSP responder. A CDP is today more of a Certificate Status Point.

The first question is which protocols must be supported. The common protocols LDAP and HTTP should be supported always. HTTP is supported by nearly all devices but some devices prefer LDAP over HTTP - especially some network solutions. Additionally there are OCSP and HTTPS. 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 not but sometimes yes. CRLs are signed by the CA and they have period of validity. There is only one working attack for CRLs. If a certificate was revoked, but the old CRL is still valid and if 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 most applications don't accept CRLs if the timestamp for the next update is over. The cool thing is that a CRL is never invalid! There can only be a newer CRL but as mentioned earlier 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 components like cables, routers, switches and servers. It is a really good idea to always have a local CDP if you only have a single interconnect to your central network. However not every branch office needs its own CDP, since if their computers (including their file and mail systems) are offline then it is not necessary to have a CDP on-line.

The third question is tricky. 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 servers one and two are not available because of a more or less big digger then your applications waits for four timeouts 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. Do you think a user would wait two minutes per recipient of an email, for verification if he wants to send a commit for a meeting?

The last question often, does not have a complete solution today. Can some supported and necessary protocols crash other applications? If you use HTTPS then you should invest some time on this problem because it is a typical chicken 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 would contact the HTTPS CDP again to verify the certificate and the verification problem would start 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. Mail servers

If you start thinking about email security then the starting point is S/MIME or PGP. However at times ethereal admins just use IMAPS and POPS. So far so good. Later they think about server based security and try to implement SMTP over TLS. Certificates issued by OpenCA can be used for SMTP over TLS because the certificates can act both as a client and a server certificate. If you use some other software than OpenCA, or you create a new role for such servers then you should read the documentation of your SMTP server really carefully.

A SMTP server can use two methods for SMTP over TLS. It can use a single 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 for the certificates 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 do not conform with RFC standards. They use the common name as a regexp to match the server name and 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, instead it checks the common name of the subject to match the DNS name or IP address. This fails if the certificate subject includes a regexp for old Netscape clients. Two non-standard compliant software packages collide here. Netscape needs regular expressions, OpenLDAP does not support regular expressions and both software packages do not check the subject alternative name first. So it's your job to make a trade-off, or to use Mozilla!

5. Organizational Aspects

5.1. Dual Access Control

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

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 conventional areas?

5.5. Enterprise Integration

already existing ERP DBs meta directories

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 end users. Perhaps you need a server CA and a separate user CA. Sometimes an old CA is still active in issuing CRLs because there are still valid certificates but the new CA issues the newer certificates. Other people use different CAs to establish an easy access control by certificate chains (so called trust paths). As you can see there are really many situations where you have to operate more than one PKI.

Most PKI programmers like me have no problem in distinguishing between different PKIs, because we always ask ourselves as to who issued the certificate. Normal users instead look at the certificate, call the hotline and ask why their certificate for Jon Doe with serial 12345 does not work. At the other end the guy from the hotline looks into his computer and answers that the certificate is correct and valid. So what's going on?

A certificate has two significant elements to identify a certificate, which are different from the common name in the subject of the certificate, and which are easy to handle. The keyID and issuer from the authority key identifier are however not easy to interpret for an end user. 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 is by using organization wide unique serial numbers. If you start a second CA or you have to replace an old CA, never reuse serial numbers if possible. You will have to search for hours if somebody calls you and reports a broken certificate chain for a certificate 12345 when you have two of those certificates. If you ever issued certificates with identical serials then always ask the issuer if you receive an error report. Never ever create a replacement for an old CA with the same name. It only causes trouble.

To sum it up in a simple manner: If you avoid duplicate identifiers then you automatically avoid many problems.

Part II. Installation and Configuration Guide

Table of Contents

Preface	xxxiii
3. Installation	34
1. Preparations	34
1.1. Software	34
1.2. Hardware	35
2. Configure	35
2.1. Host System Configuration	35
2.2. Host System Configuration (of the upcoming OpenCA 1.0)	36
2.3. Filesystem paths	36
2.4. Webserver specific stuff	37
2.5. Email	38
2.6. Compiling features	38
3. Installation	38
4. config.xml (for RPMs and DEBs too)	39
4.1. Configuration sections of config.xml	39
4.2. How to setup two management interfaces on one server?	42
4. Configuration	43
1. Access Control	43
1.1. Channel verification	44
1.2. Login	44
1.3. Session management	47
1.4. ACLs	48
2. Token and keyconfiguration	50
2.1. OpenSSL	52
2.2. Empty	52
2.3. LunaCA3	52
2.4. nCipher	53
2.5. OpenSC	59
3. OpenSSL	59
3.1. Certificate Extensions	60
3.2. Profiles	61
4. CSRs	63
4.1. Additional Attributes	63
4.2. PKCS#10 Requests	64
4.3. Basic CSR	64
4.4. SCEP	66
5. Subject	66
5.1. Common stuff	66
5.2. dc style	67
6. Subject Alternative Name	68
7. LDAP	69
7.1. Configuration of the Directory	69
7.2. Configuration of the online components	69
7.3. Writing Certificates to the Directory	71
7.4. Adding an attribute to the LDAP schema	71
8. SCEP	72
8.1. OPENCADIR/etc/servers/scep.conf	72
8.2. OPENCADIR/etc/config.xml	74
9. Dataexchange	74
9.1. Configuration	74
9.2. Adding a new node	77
10. Databases	77
10.1. PostgreSQL	77

10.2. MySQL	79
10.3. Oracle	80
10.4. DBM Files	84
10.5. SQLite	85
11. Email	85
11.1. Sendmail with basic SMTP authentication	85
12. i18n	87
12.1. Debian 3.1 Sarge	87

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 installation 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. Beginning with OpenCA 0.9.2 we no longer install all foreign modules. This is the normal behaviour of every Open Source project. The following should give you an overview about the required modules. Please note that you must install at minimum the listed version because some earlier versions like for example Net::Server include serious bugs.

Table 3.1. External Perl modules

Module	Version	Comment
Authen::SASL	2.04	required by Net::LDAP for SASL authentication - if you do not use SASL then you do not need it
CGI::Session	3.95	required for our own session handling
Convert::ASN1	0.18	???
Digest::HMAC	1.01	required by Authen::SASL
Digest::MD5	2.24	this is usually part of Perl itself
Digest::SHA1	2.02	required by OpenCA itself
Encode::Unicode	???	required by OpenCA for the internationalization stuff
IO::Socket::SSL	0.92	???
IO::stringy	2.108	???
MIME::Base64	2.20	required for Base64 encoding and decoding
MIME::Lite	3.01	required for OpenCA mail handling
MIME-tools	5.411	required for OpenCA mail handling
MailTools	1.58	required for OpenCA mail handling
Net-Server	0.86	required for OpenCA daemon - the version is important

Module	Version	Comment
URI	1.23	???
XML::Twig	3.09	used for XML parsing Warning Please read the file README in the distribution of XML::Twig which you use really carefully. There are several incompatibilities with some versions of XML::Parser and expat. The used version of Perl is heavily important too.
libintl-perl	1.10	this is our interface for the i18n stuff
perl-ldap	0.28	Perl's LDAP interface

1.2. Hardware

OpenCA was tested on several software architectures 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.

Table 3.2. Supported parameters for host configuration

Parameter	Description
<code>--with-openssl-prefix=DIR</code>	Usually OpenSSL is present on the most Unix systems because it is the best available Open Source crypto-toolkit. 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 <code>/usr/local/ssl</code> . This is the directory which you must specify. If your system already includes a proper version then you have not to use this option or you can enter <code>/usr</code> on the most Linux boxes.
<code>--with-openca-user=ARG</code>	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.
<code>--with-openca-group=ARG</code>	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 <code>openca</code> or <code>pki</code> for <i>example</i> .

2.2. Host System Configuration (of the upcoming OpenCA 1.0)

OpenCA needs two sets of user and group. One set is used for stuff which must not be changed by the OpenCA daemon and the other set is used for dynamic stuff.

2.2.1. OpenCA user and group

This is the user which owns all the directories and files which contain static stuff which must not be changed by the daemon. This user and group is a protection against software bugs and intruders.

The options are `--with-openca-user=USERNAME` and `--with-openca-group=GROUPNAME`. The most people use `root:nogroup` for this. If you have more than one PKI on one system then you should use different users and groups for different PKIs.

2.2.2. Daemon user and group

The OpenCA daemon should have an own user and group like every daemon but there is a special thing which you must take into account - OpenCA communicates via a Unix file socket. This socket has the permissions `0770`. This means that every process which is in the daemon group can write to this socket.

If you use the web interface from OpenCA then you have usually an Apache which sends data to the OpenCA socket. It is not recommended to use the same user for the Apache and the OpenCA daemon. It is recommended to put the Apache into the daemon group. This gives the Apache write access to the socket without having write access to other stuff.

2.3. 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 Perl modules 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.

2.3.1. Common Prefixes

OpenCA includes a directory structure 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.

2.3.2. 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.

2.3.3. OpenSSL prefixes (OpenCA 1.0 only)

OpenCA 1.0 requires OpenSSL 0.9.8. There are three ways of installing OpenCA with the correct OpenSSL version. First if you have installed OpenSSL in the default paths of your system then you have nothing to do. Second if you used RPM or DEB then you have nothing to do too because they use `pkg-config`. Third if you installed OpenSSL manually (e.g. in `/usr/local/ssl`) the you have to specify the path to this OpenSSL installation. The parameter is `--with-openssl-prefix=DIR`. The DIR is usually `/usr/local/ssl`.

2.4. 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 seamlessly in the server. Therefore you can configure a lot of details. So we hope you find a good tradeoff ;-)

2.4.1. 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 automatically.

2.4.2. 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 separately. 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.

2.4.3. 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.

2.5. 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.

2.6. 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 separate conflict free packages for every interfaces because all Perl modules and the common stuff can be put into separate packages.

3. Installation

First run **make**, second run **make test** 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 separately 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 includes a startup script for its 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 hundreds 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 to 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 sent mails. Usually this should be something like `<Registration Authority <pki@your_org.edu>>`. You can replace `>` by `>` but this is not required by the XML specification.

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 Technologies 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 list of 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.

<code>useLDAP</code>	If you set this option to "yes" then the LDAP code will be activated.
<code>update_ldap_automatic</code>	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.
<code>ldap_host</code>	This is the hostname of your LDAP server.
<code>ldap_port</code>	This is the port where your LDAP server listens.
<code>ldaproot</code>	The bind DN of the user which OpenCA uses to add data to the server.
<code>ldaprootpwd</code>	The passphrase for OpenCA's ldap account.

4.1.4. database configuration

If you use an OpenCA version before 0.9.3 then you have to decide which database module you want to use. OpenCA 0.9.3+ only supports SQL databases. Earlier versions support DBM files too. OpenCA versions before 0.9.3 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:

<code>db_type</code>	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.
<code>db_name</code>	name of the database which OpenCA should use
<code>db_host</code>	host of the database but sometimes the drivers don't need the host.
<code>db_port</code>	same as for host
<code>db_user</code>	the database user for OpenCA
<code>db_passwd</code>	has not to be explained :)

4.1.5. module configuration

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

Example 3.1. Module ID calculation

```
request serial ::= order number * 2^(MODULE_SHIFT) + MODULE_ID

Module ID      ::= 2
Module shift   ::= 8

order number :   real serial
-----:-----
1           : 1 * 2^8 + 2 = 258
2           : 2 * 2^8 + 2 = 514
3           : 3 * 2^8 + 2 = 770
```

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 <code>mod_ssl</code> .
SCEP_RA_CERT	This is the PEM encoded certificate of the SCEP interface. It has the same format like for <code>mod_ssl</code> .
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 <code>mod_ssl</code> .

4.1.8. 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 online components.

Additionally please remember to set the configure option `--with-node-prefix` to different names during the configuration of the online and offline installation. This is important because otherwise you have only one management interface.

4.2.1. Online Components

The first installation uses only the normal steps - `./configure --with-node-prefix=online_node -with-your-options, make, make test, 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-node-prefix=offline_node --with-your-options, make, make test, 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/offline_node  
/Test/OpenCA/lib/servers/ca  
/Test/htdocs/ca  
/Test/htdocs/offline_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 `offline_node` to `online_node`. The `cgi_prefix` must be fixed too. Please verifies the menus with the names `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 introductory 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.

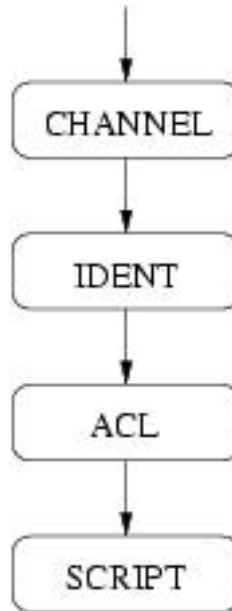
```
<openca>
  <access_control>
    The complete configuration should be here.
  </access_control>
</openca>
```

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.

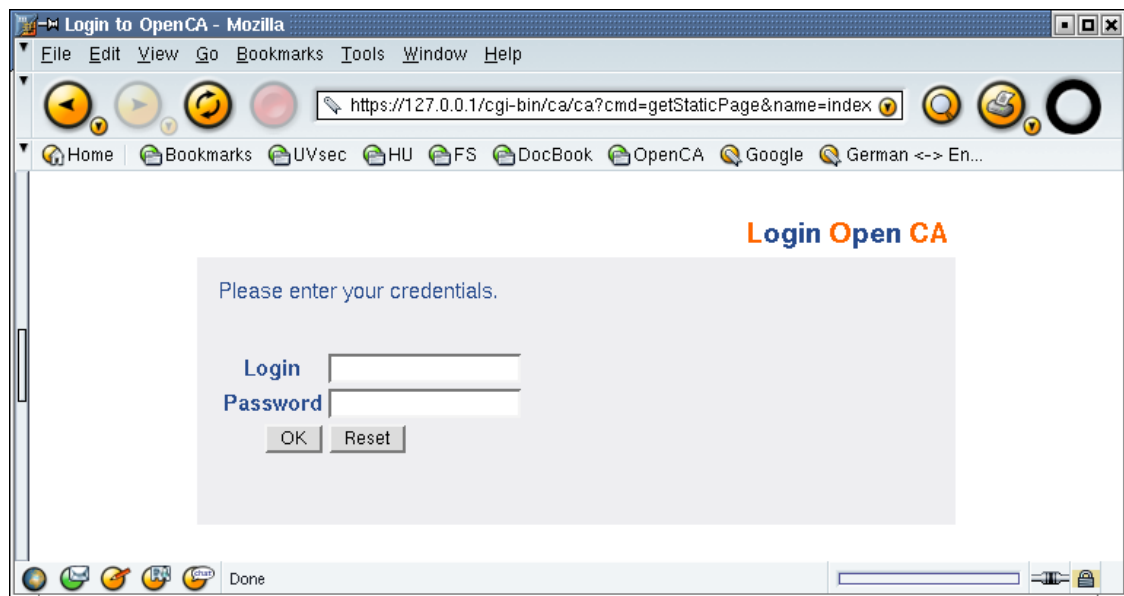
This is not only an option for debugging and testing. You can also use this option if you want to use a RA interface on a server which allows only RA access from the local machine but not over a remote computer.

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

1.2.2. passwd

This method can be used to login via login and passphrase. OpenCA supports authentication based on an internal database and based on calling an external program to perform the actual user authentication.

Figure 4.2. Passphrase based login



1.2.2.1. internal database

For the 'internal database' method you can specify one or more users. Every user has a name, a role, an algorithm and a digest. The algorithm specify which kind of digest should be used to hash the pass-

phrase. 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>
      <role>CA Operator</role>
    </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"`.

1.2.2.2. external authentication

External authentication can be used to integrate authentication methods that are not natively supported by OpenCA. Essentially this method is a variant of the username/password authentication.

External authentication requires an external program that receives the login credentials that were passed on the login screen via the environment. The external program then must verify if the username/password combination represents a valid login and return an appropriate exit code (0 for success).

The external program should take care of handling the username/password information properly, i. e. it should NOT write this information to files or pass these values to other programs on the command line. The latter is particularly important, as this might open security problems when processing untrusted user input (i. e. the login name or the password). If you must call external programs with this information, please take extra care to tidy the login information of illegal characters (such as quoting special characters). Specification of username or password as command arguments is not directory supported for exactly this reason. Use shell script wrappers that read environment variables instead.

Configuration is straightforward:

Example 4.3. External program authentication configuration

In this example configuration two environment variables `USERNAME` and `PASSWORD` are set for the external program `/usr/local/bin/authdummy`. The special strings `'__USER__'` and `'__PASSWORD__'` are replaced with the actual user login information within environment value definitions. An arbitrary number of environment variables may be defined in the configuration file.

```
<login>
  <type>passwd</type>
  <database>externalcommand</database>
```

```

    <setenv>
      <option>
        <name>USERNAME</name>
        <value>__USER__</value>
      </option>
      <option>
        <name>PASSWORD</name>
        <value>__PASSWD__</value>
      </option>
    </setenv>
    <command>/usr/local/bin/authdummy</command>
  </login>

```

As an example a very simple external authentication program `/usr/local/bin/authdummy` could (but should not) look like this:

```

#!/bin/bash
if [ "$USERNAME" = "openca" -a "$PASSWORD" = "rocks" ] ; then
    exit 0
fi
exit 1

```

Again, ALWAYS be sure to check the user input when processing the login data as arguments to external programs, it is easy to open gaping security holes here!

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 4.4. 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 4.5. 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 4.6. 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:

<code>acl</code>	enable (yes) or disable (no) the access control list - please notice that a deactivated ACL means that every user has full access to ALL OpenCA functions
<code>list</code>	defines the place of the ACL
<code>command_dir</code>	specify the directory which contains the configuration files of the scripts
<code>module_id</code>	This is the id of the interface. This id is unique for every interface.
<code>ca_cert</code>	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.
<code>map_role</code>	enable (yes) or disable (no) the mapping from certificates to roles if the certificate of the user is known. If the role is defined during passphrase based login then this option causes the use of the specified role and not the login name.
<code>map_operation</code>	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 4.7. Permission for serverInfo

```
<acl>
  <permission>
    <module>0<</module>
    <role>root<</role>
    <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 <i>map_role</i> is <i>no</i> and the role of the certificate if <i>map_role</i> is <i>yes</i> . 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 <i>map_operation</i> is <i>no</i> and the operation of the script if <i>map_role</i> is <i>yes</i> .
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 4.8. Allow all

```
<acl>
  <permission>
```

```

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

```

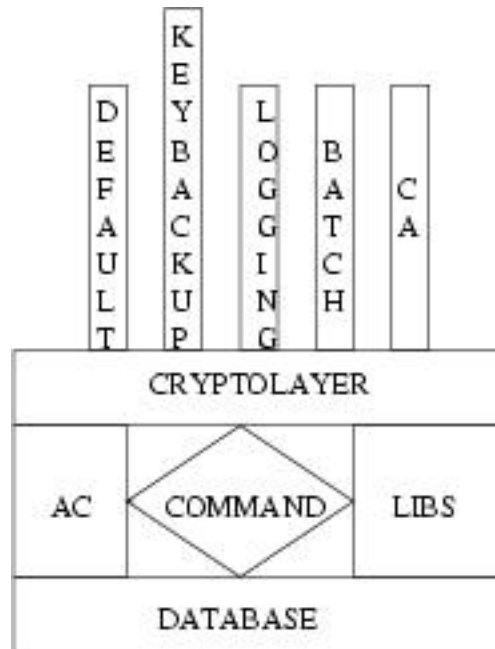
Every command or script has its own configuration file which contains the name of the command (this is actually a protection against the renaming of files), the name of the operation for which it is used, the way how to find the affected object and the name of the variable which contains the data which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

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 its 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 <i>standby</i> , <i>session</i> and <i>daemon</i> . <i>standby</i> means that you must login to the token for every action, <i>session</i> activates the token for the whole user session and <i>daemon</i> 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*. Additionally 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
PASS- WD_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 separate 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 separate 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

GENERAL LUNA CA USAGE WARNING

OpenCA knows several different internal cryptographic tokens. Such an internal cryptographic token is associated with a functionality. The CA token is associated with all CA operations. The default token is always used if no special token (or better private key) is necessary for the operation. Usually the CA token is the default token.

If you specify the CA token in `token.xml` as the `default_token` then this token is used for all normal operations. This means that the token **MUST** be activated at daemon startup. **WE STRONGLY RECOMMEND TO DO NOT DO THIS**. If you do this then all new keys are stored on the device and are usually not exportable.

We recommend that you specify a new default token which is a normal OpenSSL software token. You have not to specify a key for this token. The name of this token can be freely chosen (e.g. `default`).

It is a good idea to make the commands **hsmLogout** and **hsmLogin** visible in `menu.xml` to allow explicit logins and logouts in daemon mode.

Example 4.9. Configuration of HSM Login/logout in menu.xml

```

<item>
  <name>HSM Management<name>
  <id>2<id>
  <item>
    <name>Login to HSMs<name>
    <link>cmd=hsmLogin<link>
    <target>main<target>
  <item>
  <item>
    <name>Logout from HSMs<name>
    <link>cmd=hsmLogout<link>
    <target>main<target>
  <item>
</item>

```

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 reconnaissance 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 remember that the <i>APPID</i> must be lesser than the <i>SLOT</i> .
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.

2.4. nCipher

GENERAL NCIPHER USAGE WARNING

General warning to nCipher users (not strictly related to OpenCA).

Please note that you will always need the corresponding Security World to erase Operator Cards. You will NOT be able to erase any Operator Card once its Security World has become unavailable (i. e. /opt/nfast/kmdata lost or HSM NVRAM erased/modified AND Administrator Cards not available).

This means ***BEFORE*** removing your kmdata directory or erasing your Administrator Cards you ***MUST*** explicitly erase all Operator Cards belonging to the Security World you want to delete.

You will have to dispose of any Operator Cards that have no usable Security World any more, it is impossible

to reuse them.

2.4.1. Introduction

This module provides basic support for nCipher HSM tokens and was tested with the following configuration:

- SuSE Linux 8.1 and SLES 8
- nCipher software
 - hwsp: 0.0.24cam37, 0.0.34cam7
 - hwcrhk: 1.9.0cam27, 1.9.7cam24
 - ctls: 0.0.24cam12, 0.0.32cam27
- nCipher Modules verified (others may work)
 - nShield F3 SCSI module (nC3031S)
 - nShield F2 PCI module (nC1002P/nC3022P)
 - nShield F2 SCSI module (nC4022W)

The current status is considered to be stable.

Features:

- Uses nCipher hwcrhk application and 'with-nfast' wrapper to perform nShield private key operations. Works with OpenSSL static and dynamic engine support (automatic detection from configuration).
- Detects a number of problems and error conditions:
 - Middleware not running/operational (HSM online test)
 - Private key not available (key online test)

Current limitations:

- Key generation is not supported (and will probably never be)
- PIN entry is not supported

2.4.2. Implementation

The module was derived from OpenCA::Token::OpenSSL and other existing token implementations.

```
Error code prefix: 715*
```

```
Error codes:
```

```
7151010 Crypto Layer not defined
7151012 Token name not defined
7151013 NFAST_HOME not defined (configuration problem)
7151014 NFAST_HOME not accessible (directory does not exist or permission
denied)
7151015 Unexpected exception during program execution
7151016 PRE_ENGINE: SO_PATH not defined (configuration problem)

7153050 Key is not preloaded/usable
7153051 nCipher 'hardserver' process is not running
7153052 nCipher 'hardserver' process is not operational
7153053 Could not execute 'enquiry' program
7153054 Could not execute 'nfkminfo' program
7153055 Could not execute 'nfkmverify' program
7153056 No operational nCipher modules online
7153057 nCipher security world is not initialized / is not usable
7153058 No preloaded objects found
7153059 External program call timed out

7154001 Key generation not supported
```

All external program calls are subject to a hard timeout that is initially set to 15 seconds. This value can be configured by setting CHECKCMDTIMEOUT to the desired value in the token configuration file.

If an external program does not terminate within a certain timeout, it is killed by SIGALRM and the method fails with a timeout error (7151015). This was introduced in order to handle hanging nCipher utility processes after e. g. switching of an external SCSI HSM.

genKey() - This method is administratively disabled and always returns the error code 7154001.

online() - The online test performs the following tests:

- Run 'enquiry' and check return code and program output
 - verify that the nCipher server daemon is operational
 - verify that at least one nCipher module is operational

The method returns true without calling external checks if the module accessibility was checked within the last 60 seconds. This value can be configured by setting the variable ONLINECHECKGRACEPERIOD to the desired value in the token config file.

keyOnline() - The following tests are performed in order to determine the key online status:

- Run 'nfkminfo' with the WRAPPER command (usually 'with-nfast -M') and check return code and output.
 - verify Security World status (must be initialized and usable)
 - verify that at least one object is preloaded
 - get object hash for private key to be used and verify it against the list of preloaded objects

2.4.3. Usage

The module requires a properly set up nCipher "Security World" including a private key to operate. Key generation is not supported by the module (as it really does not make much sense for a HSM based CA).

See Section 2.4.6, "Example for the setup" for an example.

In order to use the HSM protected key, log in into the CA computer and run **/opt/nfast/bin/with-nfast pause** in a shell.

You will be prompted to insert as many Operator Cards and input their corresponding PINs as required by the Operator Card Set Quorum.

As long as 'with-nfast' is not interrupted and the last Operator Card is not removed, any application with the proper Unix permissions may use the keys protected by the Operator Card Set.

To log out from the module it is recommended to terminate the with-nfast program (pulling the Smart-Card works, too, but is not recommended, because it leaves the program waiting).

2.4.4. HSM login shell

A simple login mechanism can be implemented using a special HSM login user. The user must have write access to the file `/opt/nfast/kmdata/preload/default`. For example create a user 'hsmlogin' with primary group 'kmdata'.

Change ownership and permissions of directory `/opt/nfast/kmdata/preload` so that 'hsmlogin' can enter ("execute") this directory.

Change ownership and permissions of the file `/opt/nfast/kmdata/preload/default` so that 'hsmlogin' can write to this file and that the www user running the ca CGI script can read it.

Create a login wrapper shell script `/usr/local/bin/hsmlogin` and assign this as login shell to the 'hsmlogin' user:

```
#!/bin/sh
exec /opt/nfast/bin/with-nfast pause
```

Login as hsmlogin in order to login into the HSM, use Ctrl-C to logout.

2.4.5. OpenCA Configuration

Example 4.10. Configuration of token.xml for nCipher

```
<token>
  <name>CA</name>
  <type>nCipher</type>
<
```

```

n too          if the token support sessions then you can use session and daemo

                session - token will be logged out at end of session
                daemon  - token will be only logged out explicitly
-->
<mode>standby</mode>
<option>
  <name>DEBUG</name>
  <value>0</value>
</option>
<option>
  <name>SHELL</name>
  <value>/usr/local/ssl/bin/openssl</value>
</option>
<!-- nFast install directory (required) -->
<option>
  <name>NFAST_HOME</name>
  <value>/opt/nfast</value>
</option>
<!--
  WRAPPER defaults to '$NFAST_HOME/bin/with-nfast -M'
  You may override this default here if required, normally this
  value can be left empty.
<option>
  <name>WRAPPER</name>
  <value></value>
</option>
-->
<option>
  <name>KEY</name>
  <value>rsa-rootkey</value>
</option>
[... ]
</token>

```

The key name should be the same as reported by `/opt/nfast/bin/nfkmfinfo -k`. This configuration reflects the use of OpenSSL static engine support. In order to use dynamic engine (e. g. when using OpenSSL 0.9.8), it is necessary to specify the location of the dynamic engine nCipher shared library in the token configuration:

```

<option>
  <name>PRE_ENGINE</name>
  <value>SO_PATH:/usr/local/openssl-snap/lib/engines/libncipher.so</
</option>

```

Be sure to specify the correct location of the dynamic engine lib for the OpenSSL binary specified with the SHELL setting.

If at least one PRE_ENGINE setting is specified, the nCipher token module will automatically switch to dynamic engine semantics when talking to the OpenSSL backend. Definition of SO_PATH is the only mandatory setting in this case, and the following default PRE_ENGINE settings are automatically added (unless explicitly specified in the token configuration file):

- ID:chil
- LIST_ADD:1
- LOAD
- THREAD_LOCKING:1

2.4.6. Example for the setup

Example for creation of a Security World, Operator Cards and CA key.

Assumptions:

- example uses a 2 of 3 quorum ("2/3") for Administrator Cards and Operator Cards
- the Operator Card set protecting the CA key will be named "RootCA"
- the CA key will be named "rootkey" in this process

Sample Root Key ceremony:

1. initialize security world
 - switch HSM to 'initialize' mode
 - reset the module: **`/opt/nfast/bin/nopclearfail -c -m 1`**
 - **`/opt/nfast/bin/newworld --initialize --acs-quorum 2/3`**
 - switch HSM to 'operational' mode
 - reset the module: **`/opt/nfast/bin/nopclearfail -c -m 1`**
2. verify that the security world has been created
 - **`/opt/nfast/bin/nfkminfo`**
3. optionally: erase cards
 - **`/opt/nfast/bin/bulkerase -m 1 -s 0`**
4. initialize Root CA operator card set
 - **`/opt/nfast/bin/createocs --name=RootCA --ocs-quorum=2/3 -m 1 -s 0`**
5. verify that the operator card set has been created
 - **`/opt/nfast/bin/nfkminfo -c`**
6. create Root CA key
 - **`/opt/nfast/bin/generatekey2 --cardset=RootCA hwcrhk`** (example values: token, 0, RootCA, yes, RSA, 2048, 0, "", rootkey, no)
7. verify that the root key has been created

- `/opt/nfast/bin/nfkminfo -k`

2.5. OpenSC

This token accepts all the options like OpenSSL tokens but soem options has another meaning. The token only operates in mode *standby*. Other modes are not supported by OpenSC today. The available options have the following meanings (please notice that we only report OpenSSL options if there meaning differ from the original OpenSSL module):

KEY	The KEY is a combination of the slot and the ID of the key on the card. This can differ for other PKCS#11 modules than OpenSC. If you use this class with another PKCS#11 than the one from OpenSC then please read the docs of the vendor. The syntax is <code>slot_[0-9]+-id_[0-9]+</code> . A typical example is <code>slot_0-id_45</code> .
ENGINE	The value should be always <code>pkcs11</code> here. This defines that the OpenSSL dynamic engine <code>pkcs11</code> should be used. The module is only tested with this module.
PRE_ENGINE	These options are used to configure the PKCS#11 engine which is loaded to OpenSSL. <code>SO_PATH</code> is the path of the used OpenSSL engine. <code>ID</code> is the used OpenSSL engine ID. <code>MODULE_PATH</code> is the path to the used PKCS#11 driver. Please notice that we put the complete and partially internal configuration into this parameter. The reason is that we don't want to reduce the flexibility because we don't know the future direction of the OpenSSL engine interface and the OpenSC PKCS#11 engine for OpenSSL.
CARDDRIVER	This is the name of the OpenSC carddriver. This option is only necessary if you want to create the key with OpenCA.
CARDREADER	This is the number of the OpenSC carddriver. This option is only necessary if you want to create the key with OpenCA. (It is identical with the slotnumber.)
PKCS15_INIT	The path to the command pkcs15-init .
PKCS15_TOOL	The path to the command pkcs15-tool .
L	
OPENSC_TOOL	The path to the command opensc-tool .
L	

Note

If you get performance problems with an OpenSC engine what is normal because smartcards are not fast then please configure another default token than the CA token. You can create a new token in `token.xml` with a freely choosable name.

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 4.11. 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 4.12. Minimal SSL client extensions

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

SSL Server requires the following extensions:

Example 4.13. 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 4.14. 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 con-

figure 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 4.15. Additional attributes configuration

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

The three options have the following meanings:

Table 4.1. Additional attributes configuration

Parameter	Description
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 error detection. You can use the following values: <ul style="list-style-type: none"> • LETTERS • TEXT • NUMERIC • MIXED

Parameter	Description
	<ul style="list-style-type: none"> • 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 4.16. 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 4.17. 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

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" "Trus
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_TYPERES` 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:

Table 4.2. Generic basic CSR configuration

Parameter	Description
<code>*_ELEMENT_number</code>	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.

Parameter	Description
*_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	<p>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":</p> <p>Example 4.18. Configuration example for a XML file based HTML-select</p> <pre> <openca> <basic_csr> <basic> <element_3> <option>Computer staff</option> <option>Management</option> ... </element_3> </basic> </basic_csr> </openca> </pre>

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 *DN*s like defined by RFC 2253. There are five options which influence the subject during the issuing itself:

Table 4.3. Common stuff configuration

Parameter	Description
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 <i>Y</i> , <i>YES</i> and <i>ON</i> .
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 transparent. True values are <i>Y</i> , <i>YES</i> and <i>ON</i> .
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 <i>Y</i> , <i>YES</i> and <i>ON</i> .

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_con

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 subject 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 additional 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`. Usually it is enough to set the correct options in `ldap.xml` or in `config.xml`.

The configuration is splitted into two parts - a OpenCA related part and a LDAP related part. The OpenCA related part consists 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/`.

Supported options in interface configurations

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 configuration 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 4.19. 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 <code>ldaptls</code> and <code>ldapsasl</code> require LDAP v3. So be really careful which protocol you use. If your LDAP server supports pro-

	<p>toocol version 3 then please use it. It avoids a lot of trouble.</p>
tls	<p>Use no or yes to deactivate or activate TLS. Please remember that this option only works with LDAP v3.</p>
sasl	<p>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.</p>
excluded_roles/role	<p>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.</p>

Example 4.20. excluded roles in ldap.xml

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

7.3. Writing Certificates 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.

7.4. Adding an attribute to the LDAP schema

The common situation in large directory projects is a big schema and at the end a small request for certificate integration. This is usually no problem if OpenCA has to create and add certificates to an existent and filled LDAP server. OpenCA gets a problem if you want that OpenCA initialize this LDAP server and should create all missing nodes in the directory tree. If this is the case then you must integrate your schema specification into OpenCA's ldap configuration. You can find this schema specification in `ldpa.xml`. The path to the schema specification is `openca/ldap/schema`.

First you have to understand the general design of OpenCA's LDAP schema support. We have a very pragmatical idea of directory trees because it is not possible to handle all full featured ideas. We simply use the least significant attribute to select an appropriate schema definition for a distinguished name. Do you want to know what this mean? If you have a RFC 2253 conform distinguished name then you take the RDN on the left side. The used attribute type is used to detect the appropriate schema.

Before you can select a schema with the attribute type you must know what you want to create for a node in the directory tree. If the used distinguished name is the complete subject of a certificate then you must select the schema specification from the XML path `openca/ldap/schema/certificate`. Otherwise you have to use `openca/ldap/schema/default`. If you need to store a CA certificate then you must specify the schema in `openca/ldap/schema/ca`.

Every RDN entry in the described sections specifies the characteristics for one attributetype. This include things like required (must) or optional (may) attributes and used objectclasses (structural and auxiliary). If you have for example a new attributetype `uid_special` and a new class `MY_CLASS` then your RDN section for certificates with this attributetype as last one should looks like this.

Example 4.21. Schema extension for RDN `uid_special`

```
<rdn>
  <attributetype>uid_special<attributetype>
  <must>
    <attributetype>uid_special<attributetype>
  <must>
  <may>
    <attributetype>cn<attributetype>
    <attributetype>mail<attributetype>
    <attributetype>emailAddress<attributetype>
  <may>
  <structural>
    <objectclass>person<objectclass>
    <objectclass>organizationalPerson<objectclass>
    <objectclass>inetOrgPerson<objectclass>
    <objectclass>MY_CLASS<objectclass>
  <structural>
  <auxiliary>
    <objectclass>opencaEmailAddress<objectclass>
    <objectclass>pkiUser<objectclass>
  <auxiliary>
</rdn>
```

OpenCA's schema definition is perhaps not so flexible as you need but we are open for new ideas. Be free to mail us your ideas. If they are not too proprietary then perhaps we can integrate them :)

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`.

Note

Cisco only supports CA and end entity certificates with a keysize lower or equal 2048 bits. This means that the keysize of your CA certificate cannot exceed 2048 bits if you want to use Cisco equipment.

8.1. OPENCADIR/etc/servers/scep.conf

This file contains the following 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.
ScepAllowEnrollment	Policy definition, if set to "NO" the SCEP server will not accept requests for certificate DNs that do not exist yet. Set this value to "YES" to allow initial enrollment of new systems.
ScepAllowRenewal	Policy definition, if set to "YES" the SCEP server will allow renewal requests for existing certificates. A renewal request will not be accepted if two or more valid certificate with the same DN already exist. If set to "NO" the server will reject SCEP requests for DNs that already exist in at least one valid certificate.
ScepRenewalRDNMatch	String containing a comma separated concatenation of all RDNs to consider for matching the incoming request DN against existing valid certificate DNs. The reason for this setting is that some SCEP clients have problems generating request DNs that conform to the CA conventions. By specifying which RDNs to consider from the incoming request, it is possible to address these problems by ignoring unwanted RDNs. At least the CN should be included here. If your CNs are not unique across your name space, you should add other RDNs to create a unique reference, in most cases this would be "O", "OU" or "DC".
ScepDefaultRole	When initially enrolling a new system, this option specifies the initial certificate role to use for the new request. For renewals (at least one certificate with the same request DN already exists) the certificate role from the youngest existing predecessor is used for the new request.
ScepDefaultRA	When initially enrolling a new system, this option specifies the Registration Authority to use for the new request. For renewals (at least one certificate with the same request DN already exists) the RA setting from the youngest existing predecessor is used for the new request.
ScepAutoApprove	Newer SCEP drafts allow signing the SCEP request with a previously existing key. If a renewal request is received (this means that a valid certificate with the request DN must already exist in the database) the digital signature of this request is verified against the local CA. If the signature is valid and if the request DN is identical to the signer DN then the new request is automatically set to the 'APPROVED' status in the database. The client must explicitly support this feature by using the old certificate and private key. As the feature was introduced in later SCEP drafts, not many SCEP clients allow for automatic approval. Newer versions of scep will support this feature, however. (Please note that the client system certificate may not have the correct key usage for digitally signing a request. The SCEP draft specifically allows the violation of the key usage, and hence OpenCA's SCEP server ignores errors about incorrect key usage when verifying the signature.)

8.2. OPENCADIR/etc/config.xml

This file contains the following 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 4.22. 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 imple-

mented options to prepare and cleanup IO operations. This is necessary if you have to start special network interfaces 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 than one argument. Every argument will be separately executed. The parameters @__SRC__@ and @__DEST__@ include the directories with the data. The parameter @__DEVICE__@ will be replaced with the value of DEVICE.

Example 4.23. 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 hierarchy level LOCAL is used for backups, batch processors 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 4.24. 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__@"
EXPORT_IMPORT_DOWN_IMPORT "/usr/bin/scp openca@ra.openca.org:/usr/local/OpenCA"
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 data transfer). 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 `datexchange` 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 `datexchange`.
- 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 `datexchange`.
- 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 export. 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 database. 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 database 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 templatel
psql> create user openca with password '1234567890' createdb ncreateuser;
psql> \q
psql -d templatel -U openca
psql> create database openca;
psql> \q

```

You can restrict the database via PostgreSQL'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. Nobody else 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 to 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 migration
- 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. MySQL

Until now nobody created a howto for MySQL.

Note

If you use a version of MySQL prior 4.1 then you can see perhaps an error message which reports a syntax error because of an unknown internal variable `NAMES`. You can ignore this error message or better you should use a newer version of MySQL.

The background is the internationalization of OpenCA. We support several different character encodings. MySQL can only handle by default the character encoding which is specified for the table during

configuration. The solution is the internal variable `NAMES` which was introduced in the SQL92 standard. This variable can be used to set the actual character encoding. MySQL supports this variable beginning with MySQL 4.1. It is recommended to use at minimum 4.1.1 because of a rewrite of this stuff in MySQL.

10.3. Oracle

Oracle is supported by OpenCA via the standard Perl DBD Driver. However, there are some special considerations concerning system setup, configuration and security.

10.3.1. Perl database driver and Oracle OCI client libraries

Oracle databases are accessed via the Perl `DBD::Oracle` module which uses Oracle OCI client libraries to access database services. Both must be installed on the system.

If the location of the Oracle OCI client shared library is incorrectly configured the OpenCA daemon will not start up properly.

As a consequence, the OpenCA application must be able to locate the Oracle shared library `libclntsh.so`. Because the client library is usually not installed in a standard library path but rather below the Oracle home directory, the explicit location of this library must usually be configured on the system:

- Explicitly set the directory location of `libclntsh.so` in environment variable `LD_LIBRARY_PATH` for the OpenCA daemon. To do so you could for example write a wrapper script that calls `etc/openca_start` or include the environment setting in the `openca_start` script itself.
- Change the dynamic linker configuration by adding the required directory location to the config file (`/etc/ld.so.conf` on a Linux system). You will have to ensure that the dynamic linker configuration is updated (Linux: run the `ldconfig` command).

10.3.2. OpenCA Oracle database configuration

Note

The parameters `db_host` and `db_port` are completely ignored for Oracle databases, make sure to leave them empty in order to avoid confusion.

In `config.xml` set the `db_type` to 'Oracle'. The `db_name`, `db_user`, `db_passwd` and `db_namespace` variables may or may not be set according to your configuration (see below).

The Oracle client library requires some environment variables that must be set. As a minimum, the `ORACLE_HOME` variable must be set. This can be done via the `etc/database/DBI.conf` configuration file or by sourcing the `oraenv` file that should have been set up for your Oracle instance by your Oracle DBA.

Depending on your setup it may be useful to source the `ora_env` file in the OpenCA rc start script like in the following example. In this case, usually no environment variables need to be set up in the `DBI.conf` file.

Example 4.25. OpenCA rc script that sources Oracle environment

```

#!/bin/sh

if [ -f ~oracle/bin/oraenv ] ; then
    PATH=$PATH:/usr/local/bin
    ORAENV_ASK="NO"
    . ~oracle/bin/oraenv
fi

openca_start=/usr/local/openca-0.9.2/etc/openca_start
openca_stop=/usr/local/openca-0.9.2/etc/openca_stop

case "$1" in
    start)
        echo -n "Starting OpenCA ... "
        if $openca_start; then
            echo OK;
        else
            echo FAILED;
        fi
        ;;
    stop)
        echo "Shutting down OpenCA ... "
        $openca_stop
        ;;
    restart)
        $0 stop
        $0 start
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
esac

```

10.3.3. Internal Authentication

Internal authentication is the classic authentication method and is also the easiest to configure. It is similar to the methods used for the other database types and requires the explicit configuration of a database user, a password and a database name. The disadvantage, however, is the need to configure the database password in the configuration file which is not always desired. If this is an issue, consider the external authentication method discussed below.

The `db_user` and `db_passwd` parameters must be set to the correct username and password values for the OpenCA database. These are identical to the parameters specified in the **CREATE USER user IDENTIFIED BY "password"**; command that is used to create the user in the Oracle database.

The `db_name` parameter must be set to the database name configured in the `TNSNAMES.ORA` configuration file. As it is possible to specify aliases in this file, this may also point to an alias name of the database. In order to verify if your `db_name` is set correctly run the command **tnsping** with the `db_name` parameter as its only argument. The command must be run with the same environment variables that are set for the OpenCA daemon.

If the **tnsping** command above fails then the database setup is incorrect and must be fixed. Debugging this error is beyond the scope of this document, as there are lots of possible error causes.

If privilege separation should be used (see below) then the `db_namespace` parameter must be set to the name of the database user owning the OpenCA schema.

10.3.4. External Authentication

Internal authentication is easy to configure but has the significant drawback of requiring a cleartext password for the database user in the configuration file.

External authentication grants a Unix user access to the database without the need of a password.

This authentication type usually only works for databases running on the local machine. Although it is possible to configure external authentication for remote machines this is not recommended due to security reasons.

In order to set up an Oracle user for external authentication for the Unix user 'wwwdata' use the following Oracle command (the string 'OP\$\$' must be prefixed to the Unix user name): **CREATE USER OP\$\$WWWDATA IDENTIFIED EXTERNALLY**; After the required permissions (e. g. CREATE SESSION) are granted to the user you can test this by trying to access the database using `sqlplus /` as the user wwwdata. Make sure that the environment (in particular the `ORACLE_SID` environment variable) is set up properly for this user. The database monitor should now start without a password prompt.

To make this work with your OpenCA setup you must use the Unix user name that is configured as `daemon_user` (or `httpd_user` for OpenCA 0.9.2.x and earlier) in your `etc/openca_start` startup file.

Assuming the 'OP\$\$..' user has been created and the database schema is already set up properly, the database configuration for OpenCA must be completed.

Note

For external authentication the following configuration settings are required: `db_user` MUST be set to / (a single slash), `db_passwd` MUST be empty and `db_name` MUST be empty. In addition, the environment variable `ORACLE_SID` MUST be set properly.

10.3.5. Database privilege separation for the OpenCA application

The `db_namespace` parameter allows to specify tables that are owned by another database user. This makes it possible to create the OpenCA database scheme with one user (let's call it the schema owner) and let the OpenCA application access the tables owned by this user as another user (called the schema user). In this case the `db_user` parameter must be set to the unprivileged schema user and the `db_namespace` parameter must contain the schema owner name.

10.3.6. Sample Oracle setup

This example describes an example setup using Oracle external authentication, privilege separation and script based database setup. When used this way, the database is prepared by the setup script and NOT via the web frontend (CA/Initialization...).

Example 4.26.

This is only an example. The database schema reflects the OpenCA 0.9.2 version and is subject to change in later versions.

```
-- create schema owner
CREATE USER pkiop IDENTIFIED BY "dummy"
  DEFAULT TABLESPACE DATA01
```

```

    TEMPORARY TABLESPACE TEMP01
    QUOTA UNLIMITED ON DATA01;
GRANT UNLIMITED TABLESPACE TO pkiop;
-- schema owner does not need to log in
ALTER USER pkiop ACCOUNT LOCK;

-- create openca schema
create table pkiop.ca_certificate (ca_cert_key varchar2 (1999)
    PRIMARY KEY NOT NULL, format varchar2 (1999), data LONG,
    dn varchar2 (1999), cn varchar2 (1999), email varchar2 (1999),
    status varchar2 (1999), public_key varchar2 (1999),
    notafter number (38));

create table pkiop.crl (crl_key varchar2 (1999)
    PRIMARY KEY NOT NULL, status varchar2 (1999),
    format varchar2 (1999), data LONG, last_update varchar2 (1999),
    next_update varchar2 (1999));

create table pkiop.crr (crr_key number (38) PRIMARY KEY NOT NULL,
    cert_key number (38), submit_date varchar2 (1999),
    format varchar2 (1999), data LONG, dn varchar2 (1999),
    cn varchar2 (1999), email varchar2 (1999), ra varchar2 (1999),
    rao varchar2 (1999), status varchar2 (1999),
    reason varchar2 (1999), loa varchar2 (1999));

create table pkiop.request (req_key number (38) PRIMARY KEY NOT NULL,
    format varchar2 (1999), data LONG, dn varchar2 (1999),
    cn varchar2 (1999), email varchar2 (1999), ra varchar2 (1999),
    rao varchar2 (1999), status varchar2 (1999), role varchar2 (1999),
    public_key varchar2 (1999), scep_tid varchar2 (1999),
    loa varchar2 (1999));

create table pkiop.certificate (cert_key number (38)
    PRIMARY KEY NOT NULL, format varchar2 (1999), data LONG,
    dn varchar2 (1999), cn varchar2 (1999), email varchar2 (1999),
    status varchar2 (1999), role varchar2 (1999),
    public_key varchar2 (1999), notafter number (38),
    req_key number (38), loa varchar2 (1999));

-- create schema user
CREATE USER OPS$WWWDATA IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE DATA01
    TEMPORARY TABLESPACE TEMP01
    QUOTA UNLIMITED ON DATA01;
GRANT CREATE SESSION TO OPS$WWWDATA;
GRANT UNLIMITED TABLESPACE TO OPS$WWWDATA;

GRANT SELECT, INSERT ON pkiop.CA_CERTIFICATE TO OPS$WWWDATA;
GRANT SELECT, INSERT ON pkiop.CRL TO OPS$WWWDATA;
GRANT SELECT, UPDATE, INSERT ON pkiop.REQUEST TO OPS$WWWDATA;
GRANT SELECT, UPDATE, INSERT ON pkiop.CRR TO OPS$WWWDATA;
GRANT SELECT, UPDATE, INSERT ON pkiop.CERTIFICATE TO OPS$WWWDATA;

```

The corresponding configuration in config.xml could be (assuming the environment is properly set and the ORACLE_SID variable indicates the correct SID):

```

<!-- ===== -->
<!-- database configuration -->

```

```
<!-- ===== -->
<option>
  <name>dbmodule</name>
  <!-- you can use DB or DBI -->
  <value>DBI</value>
</option>
<option>
  <name>db_type</name>
  <value>Oracle</value>
</option>
<option>
  <name>db_name</name>
  <value></value>
</option>
<option>
  <name>db_host</name>
  <value></value>
</option>
<option>
  <name>db_port</name>
  <value></value>
</option>
<option>
  <name>db_user</name>
  <value>/</value>
</option>
<option>
  <name>db_passwd</name>
  <value></value>
</option>
<option>
  <name>db_namespace</name>
  <value>pkio</value>
</option>
```

10.4. DBM Files

Warning

DBM files are only supported in OpenCA version prior to version 0.9.3.

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 error messages 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.

10.4.1. 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
```

10.5. SQLite

Warning

SQLite is only supported in OpenCA versions after 0.9.2. The 0.9.2.x series does not support SQLite.

We replaced DBM files with SQLite. the background is simple. Both databases work out of the box without user configuration but SQLite has full SQL support. Please note that SQLite only allows one open transaction which write on the database. So SQLite is a database for a single user system. This means that you should only use it for offline machines (e.g. CA) or in test environments. Please never use it for online user interfaces.

The installation is quite simple. You have only to install DBD::SQLite. The Perl database driver includes the complete database software.

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 4.27. `/etc/mail/sendmail.mc`

```
divert(-1)dnl
```

```

dnl # Example for establishing email distribution with sendmail(client) via Microsoft Ex
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
GENERIC_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>

```

```
dn1 # => AUTH LOGIN
dn1 # mailserver replies base64 encoded, we have to reply base64-encode
dn1 # => <first user> ?
dn1 # => <then password>?
dn1 # => MAIL FROM: <valid mail-address>
dn1 # => RCPT TO: <valid mail-address>
dn1 # => DATA
dn1 # ?
dn1 # stop communication with <new line> . <new line>
dn1 # QUIT
```

12. i18n

Warning

Please note that this section applies only to OpenCA 1.0+.

If you want to use all of OpenCA locales which you can see in the submenu language then your server must support all of these locales. Nearly every OS has a different way of installing the locales. We list here only the OS where we know from some problems.

Before you start please ensure that there is no language environment if you start the server. We erase LANGUAGE from the environment but if your libc finds a language environment then the library ignores our setlocale commands and use the specified locale from the environment. This is a nice feature for client applications but a small horror for servers.

12.1. Debian 3.1 Sarge

Debian set the variable LANGUAGE by default. Sometimes this variable has some funny content (e.g. en_DE). OpenCA erases this variable automatically but it is a good idea to do it by yourself before you start the daemon.

If your OpenCA does not show some translated languages and fall back to other languages then you must activate the language on your machine. This is quite simple. You must run **dpkg-reconfigure locales** and select all necessary languages with UTF-8 encoding. OpenCA enforces the usage of UTF-8. We do not support other proprietary encodings.

Part III. User Guide

Table of Contents

Preface	xcii
5. Features	92
1. 0.10	92
2. 0.9.2	92
6. Interface Descriptions	93
1. Public PKI Server	93
1.1. General	93
1.2. CA Infos	93
1.3. User	94
1.4. Certificates	96
1.5. Requests	97
1.6. Language	97
2. Registration Authority	97
2.1. General	98
2.2. Active CSRs	98
2.3. Active CRRs	99
2.4. Information	99
2.5. Utilities	101
3. Registration Authority Node	101
3.1. General	101
3.2. Administration	102
3.3. Utilities	104
3.4. Logs	104
4. LDAP Interface	105
4.1. Update LDAP	105
4.2. View CA-Certificates	105
4.3. View Certificates	106
4.4. View CRLs	107
7. Functionality Descriptions	109
1. CA Initialization	109
1.1. Phase I: Initialize the Certification Authority	109
1.2. Phase II and III: Create the initial administrator and RA certificate	112
2. Node Initialization	113
3. CSR Handling - a request HOWTO	114
3.1. Ways to request a certificate	114
3.2. Edit a certificate signing requests	117
3.3. Approve certificate signing requests	117
3.4. Issue a certificate from a certificate signing request	117
3.5. Certificate enrollment	118
3.6. Delete certificate signing requests	118
4. Certificate Handling	118
4.1. Find a certificate	118
4.2. Download	118
4.3. Start revocation	119
4.4. Write an email to the owner	119
4.5. Informational messages and their meaning	119
5. SCEP	119
5.1. SSCEP	119
5.2. NetScreen ScreenOS	121
5.3. F-Secure VPN+	121
5.4. Cisco PIX	122
8. Client Support	123
1. Introduction	123

2. Mozilla	123
2.1. General	124
2.2. Mozilla	124
2.3. Netscape 4	125
2.4. Opera	125
3. Microsoft	125
3.1. Domaincontroller	125
3.2. Smartcard Logon	128
3.3. Keystore	129
3.4. Internet Explorer	129
3.5. Outlook	130
3.6. Outlook Express	130

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. Features

This chapter is only a short list of the features which were introduced to or removed from a version. You can use this for example to look for a special feature which you need.

1. 0.10

- DBM support was removed. Only SQL databases are supported
- You can hide certificates of a special role.
- Serial generation via sequence generators of databases.
- No memory leaks during batch operation.
-

2. 0.9.2

- Full stylesheet support.
- nCipher support.
- OpenCA daemon was introduced.
- XML cache was introduced.
-

Chapter 6. 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.

There are a set of "tabs" along the top of the screen, each is described below.

1.1. General

This section describes the current versions of the OpenCA modules. There is only one sub menu item, Logout.

1.1.1. Logout

This link has no function as the user has not logged onto the Public Interface.

1.2. CA Infos

This section describes the CA related utilities a user can access. Each heading below relates to a link under the CA Infos tab.

1.2.1. Policy

This link displays the CA policy as a web page.

1.2.2. Get CA Certificate

By hitting this link the user is presented with a page titled "Download and Install CA Certificates". 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 in format PEM" to download the certificate in the appropriate format for inclusion in the Apache configuration files.

1.2.3. Certificate Revocation Lists

By hitting this link the user is presented with a screen entitled "Download and Install CRLs". 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.

1.3. User

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

1.3.1. Request a certificate

This link presents the user with a page offering a number of certificate request methods. There are subtle 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
- Alternative email: Another email to appear in the certificate
- DNS name: The DNS name if the certificate is a web server cert
- Name: The real name of the user
- Email: The users email address
- Department: The user's department
- Telephone: The users telephone number
- Level of Assurance: This is the type of physical authentication the user is to receive
- 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 needs to revoke their own certificate in the future.

1.3.1.1. 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 certi-

ificate. 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.

1.3.1.2. 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, or if for some reason they wanted the Certificate Authority to generate the keys and CSR (e.g. For key backup on the server).

1.3.1.3. Netscape's Request

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

1.3.1.4. IE Request

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

1.3.1.5. 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.

1.3.1.6. 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. Sincerely your Jon Doe".

1.3.2. 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.

1.3.3. 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.

1.3.4. 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.

1.4. 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.

1.4.1. Valid

Following this link the user 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).

1.4.2. 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.

1.4.3. 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.

1.4.4. 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.

1.4.5. 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.

1.5. Requests

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

1.5.1. 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.

1.5.2. 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.

1.6. Language

This heading lists the languages that OpenCA has available, by clicking on one of the language links, the screens change to the selected language.

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.

The user is first asked to authenticate themselves to the RA, depending on the configuration, this authentication may be nothing, username and password or by certificate.

Each one of the headings below corresponds to tab across the top of the default RA screens.

2.1. General

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).

2.1.3. Logout

Pressing this link logs the user out of the interface.

2.2. Active CSRs

This tab list functions that can be performed on Active Certificate Signing Requests, i.e. requests from users for a certificate.

2.2.1. New

This link shows new CSRs at a specific Registration Authority with a certain Level of Assurance (as specified by the user at certificate request time). The RA Operator chooses the RA and LoA.

The screen shows all the new CSRs.

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

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 unnecessary.

2.2.1.4. Delete Request

Pressing this button deletes the request from the system.

2.2.2. Renewed

This screen displays any re-newed certificate signing requests. The list of options is the same as the "New" function.

2.2.3. Pending (be processed already)

This screen displays any already processed CSR's

2.2.4. Waiting for additional signature

In some circumstances CSR's require two signatures, those requests are displayed here. The functionality is the same as "New" requests.

2.3. Active CRRs

A user can initiate their own certificate revocation or it can be initiated by an RA Operator. This screen shows Certificate Revocation Requests in various states.

2.3.1. New

This section shows new certificate revocation requests. The RA Operator can process them by clicking on the CRR serial number.

2.3.1.1. Approve and Sign Request

Pressing this button allows the RA User to approve the revocation 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).

2.3.1.2. Approve Request without Signing

Pressing this button approves the revocation 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 unnecessary.

2.3.1.3. Delete Request

Pressing this button deletes the revocation request from the system.

2.3.2. Pending (be processed already)>

This section shows CRRs that have been approved and exported to the CA.

2.3.3. Waiting for additional signature

Some CRRs require two signatures before they can be processed, these are displayed here. The RA Operator process them like "New" requests.

2.4. Information

This tab allows the RA Operator a different view of CSRs, CRR, User certificates, CA certificates and CRLs.

2.4.1. Certificate Requests

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

The following lists of certificate requests can be displayed.

- New
- Renewed
- Pending
- Signed (waiting for additional signature)
- Approved
- Archived
- Deleted

2.4.2. Revocation Requests

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

The following lists of certificate revocation requests can be displayed.

- New
- Pending
- Signed (waiting for additional signature)
- Approved
- Archived
- Deleted

2.4.3. Certificates

This link displays information about certificates in the PKI.

The following lists of certificates can be displayed.

- Valid
- Expired
- Suspended
- Revoked

2.4.4. CA Certificates

This link displays information about CA certificates in the PKI.

The following lists of CA certificates can be displayed.

- Valid
- Expired

2.4.5. CRLs

This link displays information about CRLs in the PKI.

The following lists of CRLs can be displayed.

- All

2.5. Utilities

This section contains RA Operator utilities.

2.5.1. Search Certificate

This allows the RA Operator to search for a specific certificate based on Name, Email, DN or Role.

2.5.2. Search CSR

This allows the RA Operator to search for a specific certificate signing request based on Name, Email, DN or Role.

2.5.3. Warn Expiring Certificates

This allows the RA Operator to search for certificates expiring in the next "N" days (default 31).

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. General

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 accessible. 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 Admin

This section takes the user to a screen to manage the LDAP server, if one has been configured.

3.1.4. Public

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

3.1.5. Logout

Logs the user out of OpenCA.

3.2. Administration

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

3.2.1. Stop Daemons of Crypto Tokens

If the PKI has been configured with Crypto Tokens holding the CA private key in an online mode, then this link will stop the token daemon.

3.2.2. 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:

3.2.2.1. 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 lose all existing data. Be careful !

3.2.2.2. 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.

3.2.3. 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.

3.2.3.1. Enroll data to a lower level of the hierarchy

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

3.2.3.2. Receive data from a lower level of the hierarchy

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

3.2.3.3. 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).

3.2.3.3.1. All

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

3.2.3.3.2. Certificates

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

3.2.3.3.3. CRLs

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

3.2.3.3.4. Configuration

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

3.2.3.3.5. Batchprocessors

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

3.2.3.4. 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.

3.2.3.4.1. All

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

3.2.3.4.2. Requests

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

3.2.3.4.3. CRRs

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

3.2.4. 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.

3.2.4.1. 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).

3.2.4.2. 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.

3.2.4.3. Restore Database

Use this link to recover the backup data.

3.2.4.4. 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.

3.2.5. Database

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

General utilities for the RA Operator

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. Cleanup sessions

I don't know what this does !!!???

3.3.4. Delete Temp Files

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

3.3.5. Rebuild CA Chain

This link runs a function that rebuilds the CA Chain, this is useful if your have a CA hierachy and need to tell the environment about the chain of CA certificates.

3.4. Logs

Utilities to handle log files.

3.4.1. Search

This function lets the RA Operator search the log files for log entries of a specific class and level.

3.4.2. Recovery index database

I do not know what this does !!!

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 directory 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

Writes 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 are 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 are 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 7. 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 7.1. Phases of the CA initialization

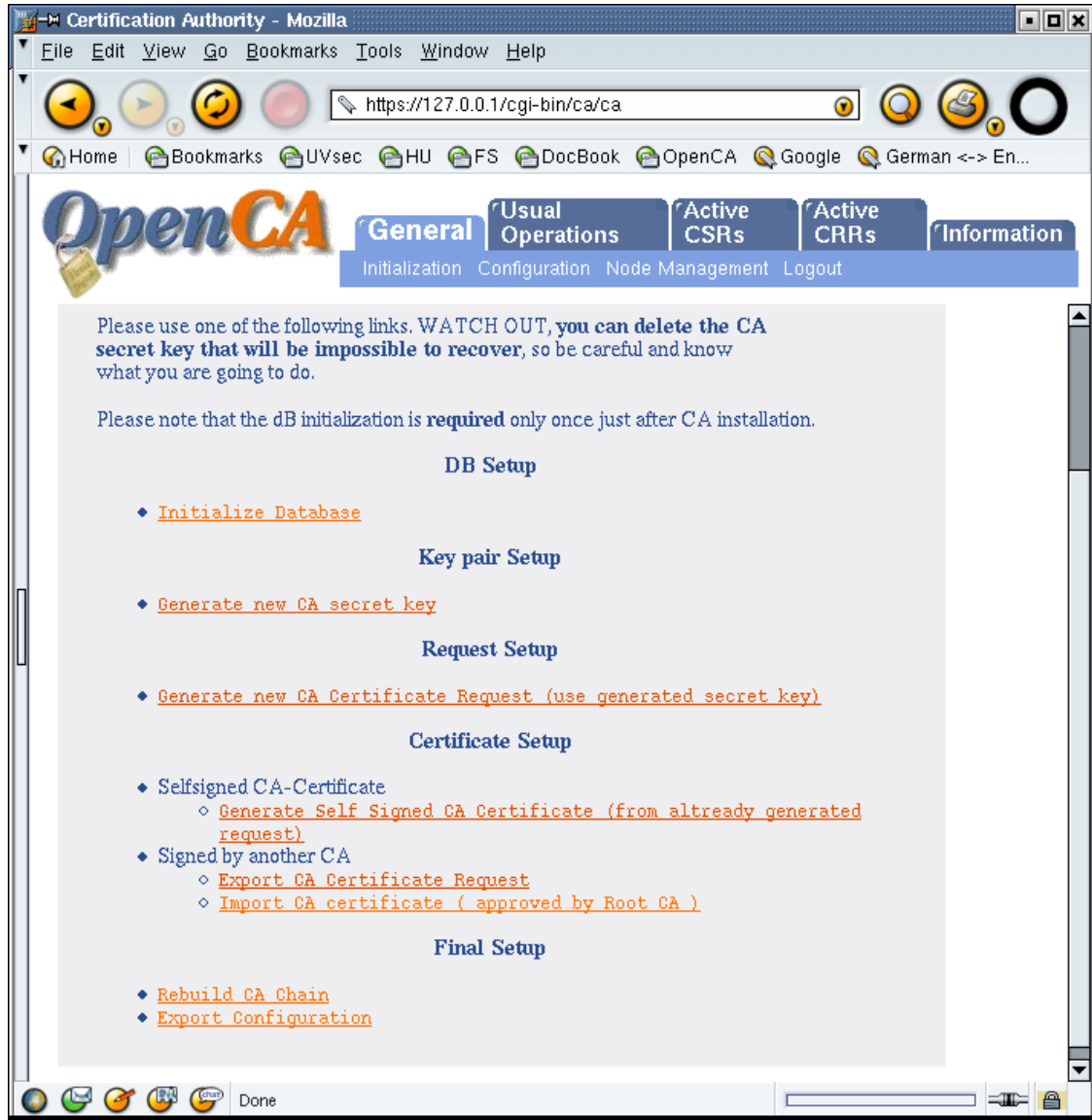


1.1. Phase I: Initialize the Certification Authority

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 7.2. 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. Please note that OpenCA 0.9.3+ only supports SQL databases.

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 parameters 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 the 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 initialization 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 your 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 phases are used to create the first certificates of the infrastructure. You can create these certificates via a separate 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 7.3. Phase II of the CA initialization

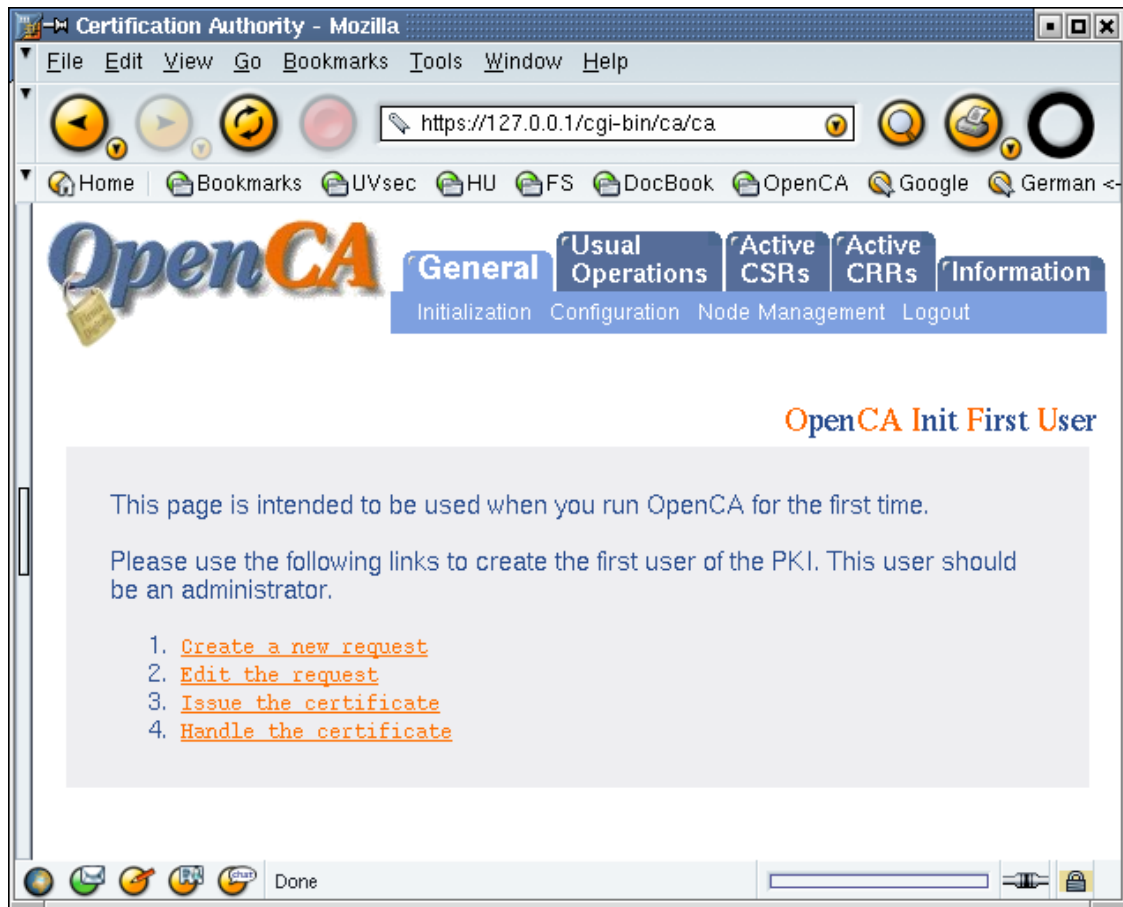
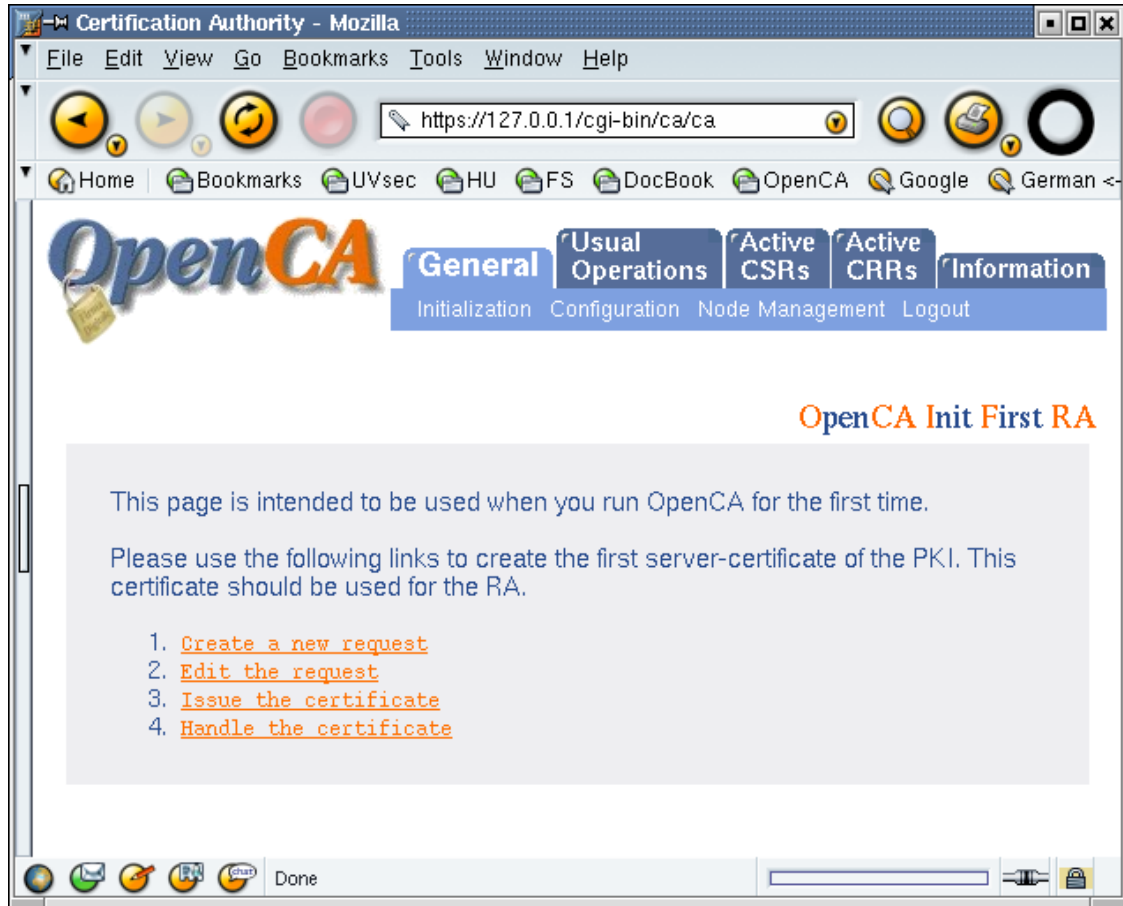


Figure 7.4. Phase III of the CA initialization



Both phases use the same process like for every normal certificate signing request. You can simply read the user guide section where the different requests and their handling will be described. The phases consist 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 want to setup one or more online servers to publish certific-

ates 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 certificate 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. There are a number of restrictions to the content of the subject fields, do not use characters "#", "," and ";".
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 to delete the passphrase after a short period of time and to generate the passphrases with things like `openssl rand`. User or admin “generated” passphrases 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 your 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 activated 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 7.1. 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. You can do this with the command **sscep getca -f scep.conf**. 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 scep.conf**. The result is some debugging output and scep 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.

Warning

Please notice that F-Secure has a problem with DNS. If your VPN+ does not work then it is a good idea to replace the domain name in the URL with the IP of the server.

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 <i>Only in SubjectAltName extension</i> 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 email-address, 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>].

5.4. Cisco PIX

The most OpenCA SCEP users are using PIX but nobody writes a good documentation until now :(We can only notice that it is sometimes necessary to create a new role which only includes the extension for the Netscape certificate type. It looks like the PIX is some kind of sensible for special extensions.

Please notice that the PIX has one major problem - it cannot scale for large installations. This does not mean that there is not enough performance but the box can only handle one CA. This mean that if your have to replace your old CA by a new one then the trust for all old certs breaks immediatly. It is not possible to operate with two CAs at the same time. If you have a large company with thousands of employees then it is nearly impossible to replace all user certs at one time.

Timestamp: 2004-May-17

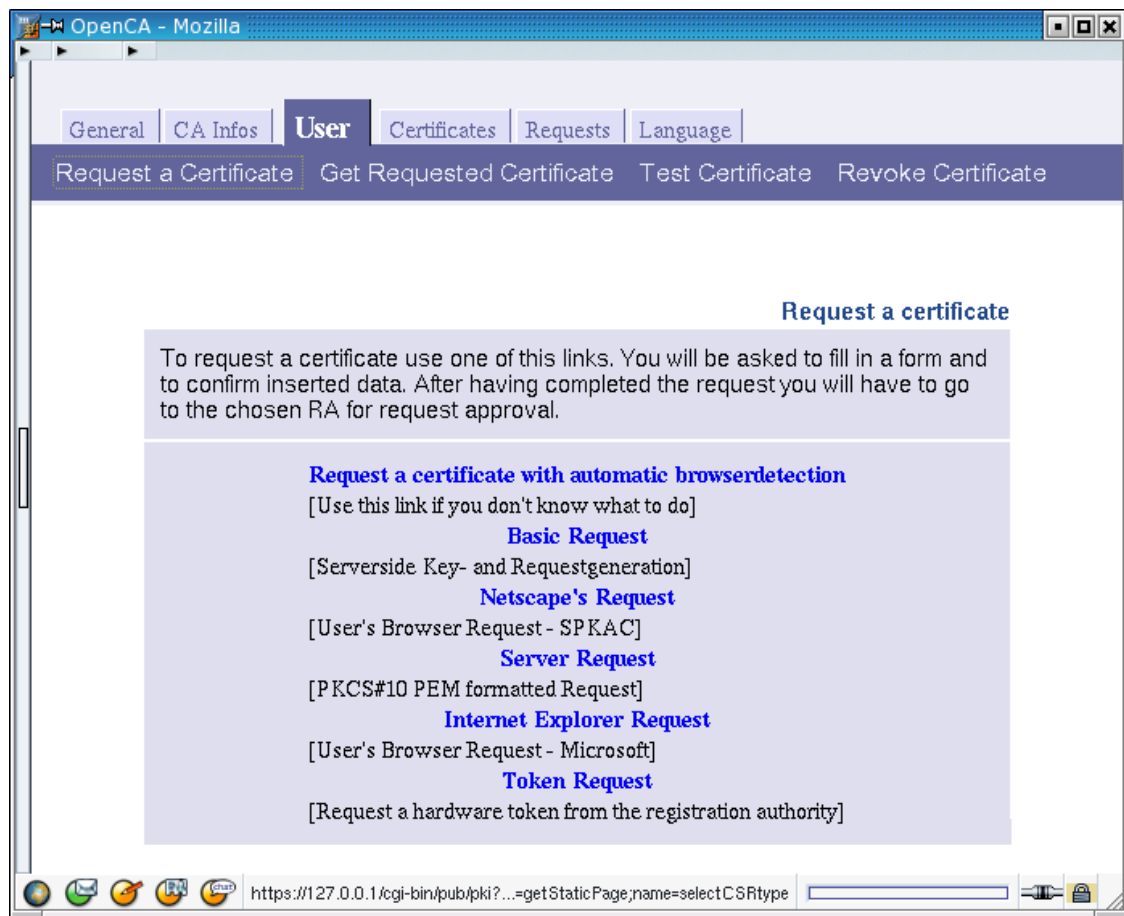
Chapter 8. Client Support

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 8.1. 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 click onto the User area. There you can choose an option `request a certificate`. After this you can choose between several options (see Figure 8.1, "Request a certificate").

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 information 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

There were several problems with Mozilla and signing in the past. The most problems should be fixed if you are using Mozilla 1.7+ or Firefox 0.9.3+.

2.2.2.1. Mozilla 1.7+ and Firefox 0.9.3+

These versions of Mozilla support the same technology like the old Section 2.3, "Netscape 4". You must have the complete CA chain of a certificate and you must trust the CA certificate. If these requirements are meet then you can use a certificate to sign an HTML form. This is completely done in Javascript. No additional plugins are required.

2.2.2.2. Mozilla 1.0 to 1.6, Firefox up to 0.9.2 and Netscape 6 and 7

If you want to sign HTML forms with the old versions of the Mozilla browser then you must use Section 2.2.2.3, "SecClab".

2.2.2.3. SecClab

Secclab is a XPCOM Component that implements some PKI functions. This sounds a little bit abstract but it is the official description from <http://secclab.mozdev.org>. In fact SecCLAB is a plugin which implements form signing.

All Mozillas from 1.0 to 1.6 and the modern Netscape version doesn't include the old crypto object from Netscape 4. This statement is correct for all Firefox up to and including 0.9.2 because Firefox uses the core code from Mozilla. The result is that you are unable to sign forms with these browsers. OpenCA needs this functionality to protect approved requests and to support X.509 based authentication. The plugin implements a new object class which replaces the old function `signForm`.

If you want to download or install the plugin from the homepage of SecCLAB then please notice that you must take care about the correct version. There are especially for Linux two different versions - one for gcc3 and one for the old binaries. All new distributions use gcc3. Debian woody uses 2.95 and 2.96 (ia64). Only hppa (HP PA-RISC) uses gcc 3.0 but the plugin was only compiled for i386. After the installation you have to quit Mozilla and start again. Please really quit Mozilla and not only close the used window (usually Ctrl-Q do the job). If you don't find the appropriate plugin in the download area then please check the installation area. There was some confusion with the gcc3 version for Linux.

If you want to sign something with a certificate then you must install the complete CA chain and you must trust the CA. The important thing is that you must trust the CA for email signing. We do not know what the signing of some data has to do with email signing but without this it does not work.

Note

It is strongly recommended to not use SecCLAB any longer because the actual Mozilla (1.7+) and Firefox (0.9.3+) versions include support for formsigning. SecCLAB was only necessary until the development team of Mozilla implemented the old feature from Netscape 4.

2.2.2.4. WaMCom

WaMCom should no longer be used because the actual versions of Firefox and Mozilla include all required components now include an up-to-date set of security patches.

2.3. Netscape 4

2.3.1. Backup a certificate

2.3.2. Signing Data

You must have the complete CA chain of a certificate and you must trust the CA certificate. If these requirements are met then you can use a certificate to sign an HTML form. This is completely done in Javascript. No additional plugins are required.

2.4. Opera

3. Microsoft

3.1. Domaincontroller

Before the description starts please notice that the SMTP based replication of Microsoft ADS doesn't work with *TTPs*. Microsoft officially commits this problem but doesn't comment the reasons. If you don't use SMTP replication then you can TTPs without any problems except that Microsoft's automatic certificate renewal does not work of course.

The requirements are described in detail in the knowledge base article 291010 [<http://support.microsoft.com/default.aspx?scid=kb;en-us;291010>]. The most important things are listed here:

- There must be at minimum one CDP.
- The X.509v3 extension Key Usage must include the options Digital Signature and Key Encipherment.
- The X.509v3 extension Extended Key Usage must include Client Authentication (1.3.6.1.5.5.7.3.2) and Server Authentication (1.3.6.1.5.5.7.3.1). This is necessary because domain controllers will be connected as servers and establish connections as clients to other servers.
- The X.509v3 extension Subject Alternative Name must include the globally unique identifier (GUID) of the domain controller and the DNS name of the domain controller. The GUID is a 16 Byte long number which identify the object of the domain controller in the directory (ADS). The OID of the GUID in the othername is 1.3.6.1.4.1.311.25.1.
- There must be one extension with the BMP value DomainController inside. Microsoft has an extra extension for this template name with the OID 1.3.6.1.4.1.311.20.2. The extension is referenced by Microsoft as Certificate Template Name.
- The X.509v3 extension Basic Constraints should include End Entity as subject (OpenSSL: CA=false) and no limitation for the path length.
- The subject of the certificate can be the directory path of the domain controller. This is only optional.

If you want to create such a certificate then you must create the key pair with 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.

3.1.1. OpenSSL 0.9.7

The settings for Key Usage, Extended Key Usage and the CDPs is no problem with OpenSSL and we don't describe here in detail how you can configure this in OPENCADIR/etc/openssl/extfiles/domaincontroller.ext. The Basic Constraint and Subject stuff is the same like for every other certificate too.

Example 8.1. OpenSSL 0.9.7 key usage and extended key usage for DCs

```
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, serverAuth
```

The Subject Alternative Name is the first difficult thing. OpenSSL 0.9.7 does not support othername. The solution is to encode the GUID in binary format into the subject alternative name.

Example 8.2. OpenSSL 0.9.7 subjectAltName for DCs

```
2.5.29.17=DER:30:32:A0:1F:06:09:2B:06:01:04:01:82:37:19:01:A0:\
12:04:10:01:02:03:04:05:06:07:08:09:10:11:12:13:14:15:16:82:0F:\
64:6E:73:2E:73:6F:6D:65:68:6F:73:74:2E:64:65
```

You can find a detailed description of DER: in `openssl.txt` of the OpenSSL documentation. 2.5.29.17 is the OID of the subject alternative name. The example GUID is 01020304050607080910111213141516. Please remember that the GUID is always a 16-byte string and the numbers are hexadecimal encoded. A 16 is a decimal 22. The DNS name is `dns.somehost.de` (64:6E:73:2E:73:6F:6D:65:68:6F:73:74:2E:64:65). If you need more infos about ASN.1 encodings then please read the ASN.1 specs. We only know that the 10 in front of the GUID and the 0F in front of the DNS name are the length of the values.

The next complicated stuff is the certificate template name but this is a fixed extension so we can copy it from already existing certificates :)

Example 8.3. OpenSSL 0.9.7 certificate template name for DCs

```
# 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
```

After you read all these complicated issues you understand perhaps why I can only recommend you to use OpenSSL 0.9.8 if you want to issue certificates for domain controllers. The biggest problem is the subject alternative name which can only be created in binary format which requires to change the extension configuration file of the used role for every issued certificate to encode the correct GUID and DNS name of the domain controller. Nevertheless you can do it with OpenSSL 0.9.7.

3.1.2. OpenSSL 0.9.8

The first important note before you start with OpenSSL 0.9.8 - you cannot compile OpenCA with OpenSSL 0.9.8. You must install OpenCA with an OpenSSL 0.9.7 and then you must change the path of the OpenSSL binary from the 0.9.7 binary to the 0.9.8 binary. We know that this is not really comfortable but the header files changed from 0.9.7 to 0.9.8 in an incompatible way and we only migrate our sources if there is a 0.9.8 stable release.

The standard things like Key Usage, Extended Key Usage, Subject, CDPs and Basic Constraints do not change between the different OpenSSL versions. The certificate template name extension of Microsoft must be copied as binary too but it is like the other standardized extensions a static string.

The important improvement of OpenSSL 0.9.8 is the support for `othername` in the subject alternative

name. The new OpenSSL version allows a much easier specification of proprietary extensions even if they are in binary format.

Example 8.4. OpenSSL 0.9.8 subject alternative name section for DCs

```
subjectAltName=@altname_sec
[ altname_sec ]
otherName=1.3.6.1.4.1.311.25.1;FORMAT:HEX,OCT:01020304050607080910111213141516
DNS=dns.somehost.de
```

The GUID is the same GUID like in the example for OpenSSL 0.9.7. The big question is now how can I put it into my OpenCA? The answer is short. You have to choose otherName as attribute for the subject alternative name at the RA interface and then you have to enter 1.3.6.1.4.1.311.25.1;FORMAT:HEX,OCT:01020304050607080910111213141516.

We know that it is a big problem for the most admins to enter OIDs and cryptical things like OpenSSL formats. Therefore we added some extra stuff for Microsoft only. You can use the field MS_GUID as a subject alternative name attribute too. There you have only to enter the pure GUID in HEX format. The rest is handled by OpenCA.

3.2. Smartcard Logon

The requirements are described in detail in the knowledge base article 281245 [<http://support.microsoft.com/default.aspx?scid=kb;en-us;281245>]. Please note that we only list the requirements for the PKI system here. There are several other requirements to which you can find in the knowledge base article. The most important things are listed here:

- There must be at minimum one CDP.
- The X.509v3 extension Key Usage must include the options Digital Signature.
- The X.509v3 extension Extended Key Usage must include Client Authentication (1.3.6.1.5.5.7.3.2) and Smart Card Logon (1.3.6.1.4.1.311.20.2.2).
- The X.509v3 extension Subject Alternative Name must include the universal principal name (UPN) of the user. A UPN is like account@domain (e.g. john_doe@company.com). It is important to understand that this is no emailaddress. The OID of the UPN is 1.3.6.1.4.1.311.20.2.3. The format of the UPN is UTF-8.
- The X.509v3 extension Basic Constraints should include End Entity as subject (OpenSSL: CA=false) and no limitation for the path length.

3.2.1. OpenSSL 0.9.7 (patched)

Please use the patch for OpenSSL which you can find at our ftp server. The patch othername.tgz includes a fix for Microsoft's othername usage. The following documentation only refers to this patched version of OpenSSL.

OpenCA's role `User` already includes the required extensions for Microsoft's smartcard logon. Generally you can add the OID for secure email (1.3.6.1.5.5.7.3.4) and other things to such a certificate. The smartcard certificate is not limited to logon. To be more Microsoft-like you can add a certificate template name to the certificates like for domain controllers. This is not required but it looks more nice :)

Example 8.5. `extendedKeyUsage` for clients

```
# 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
```

Now we have to deal with the UPN. This UPN must be placed in the `othername` of the subject alternative name. If you use the patched option and the `othername` is supported by the RA configuration then you have to enter `1.3.6.1.4.1.311.20.2.3:UTF8String:account@domain` in the value field after you selected `otherName`. The keyword `UTF8String` is case-sensitive. If you do all correct then you can now issue a wonderful certificate for smartcard logon.

3.2.2. OpenSSL 0.9.8

The only difference between OpenSSL 0.9.8 and 0.9.7 is the handling of the `othername`. OpenSSL 0.9.8 support this by default without patching. You have to enter `1.3.6.1.4.1.311.20.2.3;UTF8:account@domain`. Please notice that OpenSSL 0.9.8 uses `UTF8` and not `UTF8String`.

We know that it is a big problem for normal users to enter OIDs and other cryptical things like which have nothing to do with the UPN. Therefore we added some extra stuff for Microsoft only. You can use the field `MS_UPN` as a subject alternative name attribute too. There you have only to enter the pure UPN like an emailaddress. The conversion to OpenSSL's format is handled by OpenCA.

3.3. Keystore

The private keys of a user are stored in the profile of the user if no smartcards are used. This means that the private keys of any Microsoft client software like Internet Explorer or Outlook are automatically present after a successful login because they are part of the profile. This protects the keys too.

3.4. Internet Explorer

3.4.1. Requesting a certificate

3.4.2. Installing a certificate

3.4.3. Backup a certificate

3.4.4. Signing Data

3.5. Outlook

3.6. Outlook Express

Part IV. Technology Guide

Table of Contents

Preface	cxxxiv
9. Introduction	135
1. Slottechnology	135
10. XML	137
11. Cryptolayer	138
12. Accesscontrol	140
13. Logging	145
14. Webinterfaces	147
1. Interfacebuilding	147
1.1. Technology overview	147
1.2. Customization capabilities	149
2. CSS	150
3. Configuration after installation	150
15. Hierarchy	151
1. Nodemanagement	151
2. Dataexchange	151
16. LDAP	152
1. LDAP schema specification	152
1.1. Used objectclasses	152
1.2. Supported attributes	152
1.3. Common definitions for distinguished names	153
1.4. Special definitions for user certificates	154
2. Sourcecodeorganization	154
2.1. Structure of the code	154
2.2. The relevant commands	155
2.3. export-import.lib	155
2.4. ldap-utils.lib	155
2.5. OpenCA::LDAP	155
17. Database	156
1. Tables	156
2. Sequences	157
3. Indexes	157
18. Batch System	158
1. Requirements	158
2. Design	158
3. Data Import	159
4. Database background	161
5. Change the workflow	162
6. Default workflow	163
7. What about the different crypto tokens?	164
8. Performance	164
8.1. PIII 850MHz, 256 MB RAM	164
19. Packaging	165
1. Common Notices	165
1.1. Required Perl modules	165
2. RPM-based system	165
2.1. RedHat/Feodora	165
2.2. SuSE	165
3. Debian	167
4. BSD	167
20. Software Design (legacy from design guide)	168
1. Database(s)	168
2. Interface construction	168

3. openca.cgi	168
4. libraries	168
5. modules	168
6. commands	168
7. Dataexchange and Node management	168

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 9. 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 evolutionary 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 evolutionary 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 necessarily 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 10, *XML*
- Chapter 11, *Cryptolayer*
- Chapter 12, *Accesscontrol*
- Chapter 13, *Logging*
- Chapter 14, *Webinterfaces*
- Chapter 15, *Hierarchy*
- Chapter 16, *LDAP*
- Chapter 18, *Batch System*
- Chapter 17, *Database*

1. Slottechnology

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 10. 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.

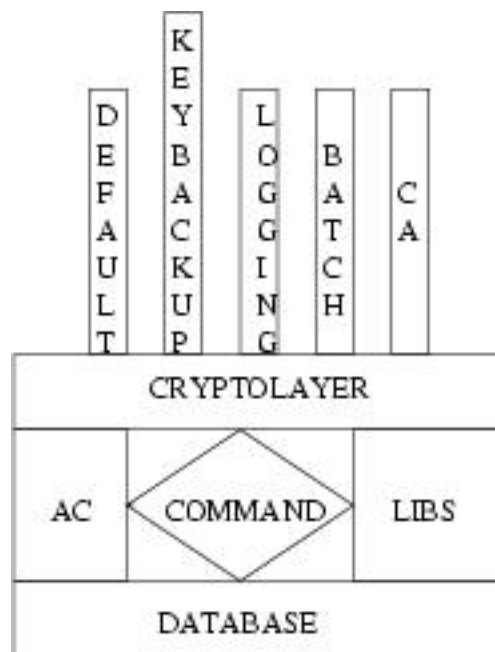
Today we have no DTD but it **MUST** be written before version 1.0.

Chapter 11. 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, "Slottechnology" to manage all the different keys. There is a central class `OpenCA::Crypto`. This class can load and manage so called "tokens". The software itself uses at every time a special token. Usually there is a default token `$CryptoShell` available which is not associated to a special key.

Figure 11.1. Example cryptolayer with tokens



If a script or function needs a special key then it calls `getToken` and 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 implement 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 are two additional classes beside the classes for empty and software keys - `LunaCA3` and `OpenSC`. The class `LunaCA3` can manage SafeNet LunaCA3 and Luna SA (former Rainbow former Chrysalis-ITS) devices. 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 wants. The daemon mode activates the HSM and runs it until there is an explicit shutdown for the HSM via an interface. The class `OpenSC` can use any `OpenSC` smartcard as a HSM but it can also be used to integrate every other PKCS#11 device if you don't use the keygeneration of this class. The keygeneration is specific for `OpenSC`. The other stuff

bases on OpenSC's PKCS#11 engine for OpenSSL which uses the PKCS#11 module of OpenSC. You can replace OpenSC's PKCS#11 library by any other PKCS#11 library.

Chapter 12. 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 configuration and to the access control list if such a list will be used. The file itself has the usual OpenCA structure:

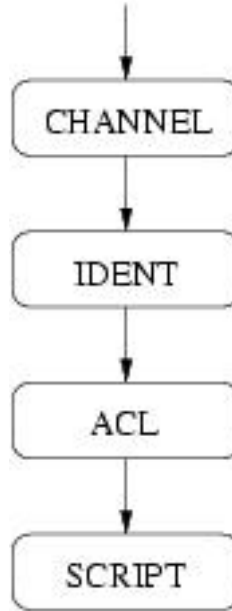
```
<openca>
  <access_control>
    The complete configuration should be here.
  </access_control>
</openca>
```

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, "Slototechnology" 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 12.1. Passes of the accesscontrol



The different passes works like follows.

Figure 12.2. Channel verification

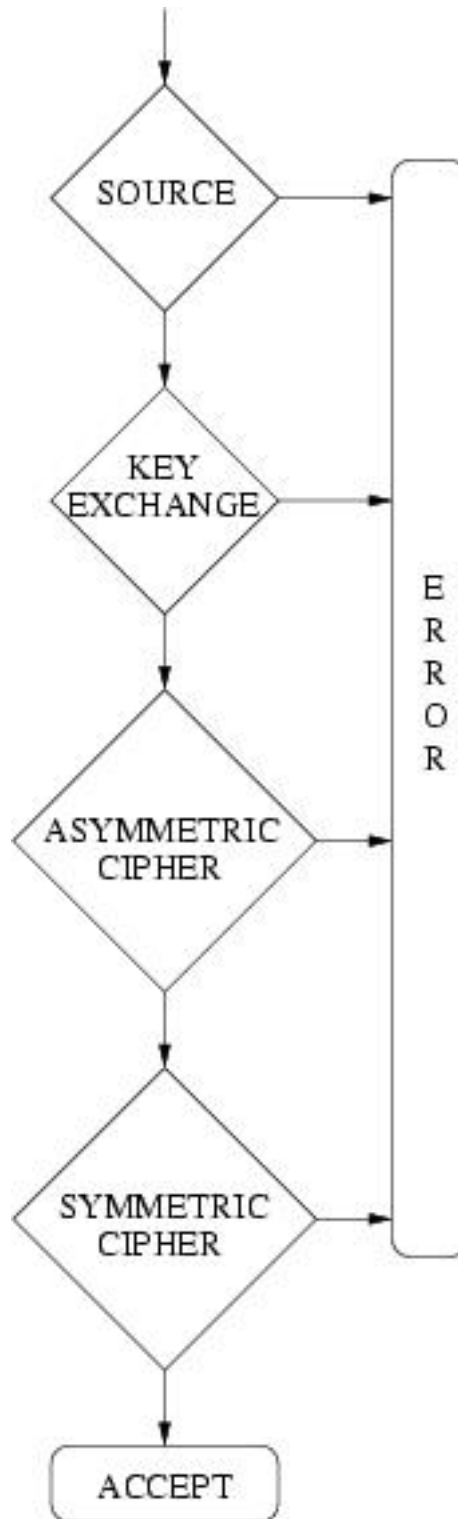


Figure 12.3. Identification of the user

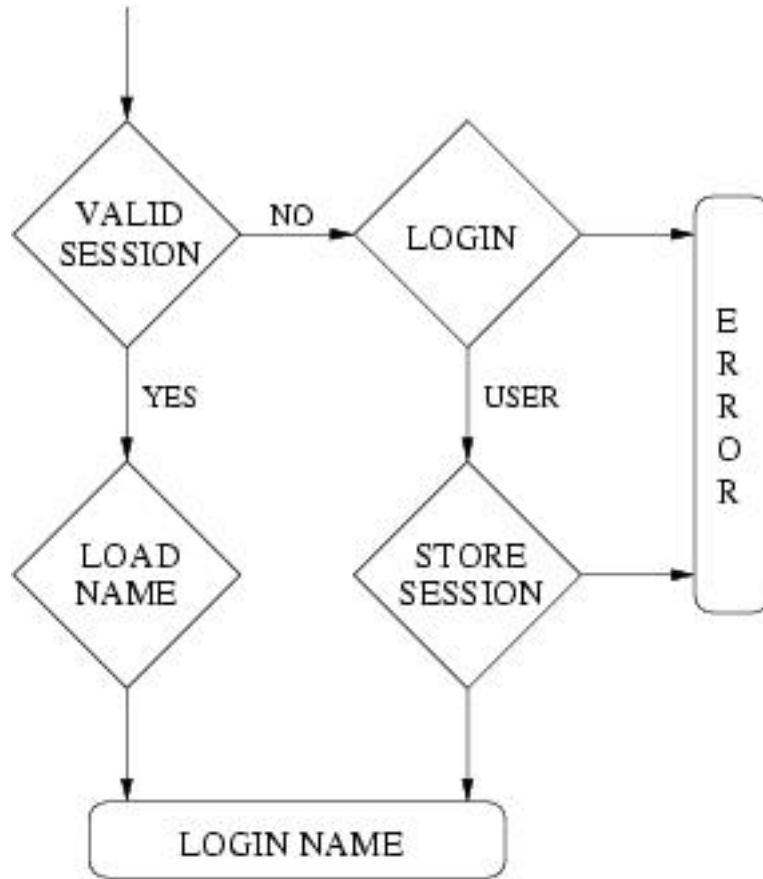
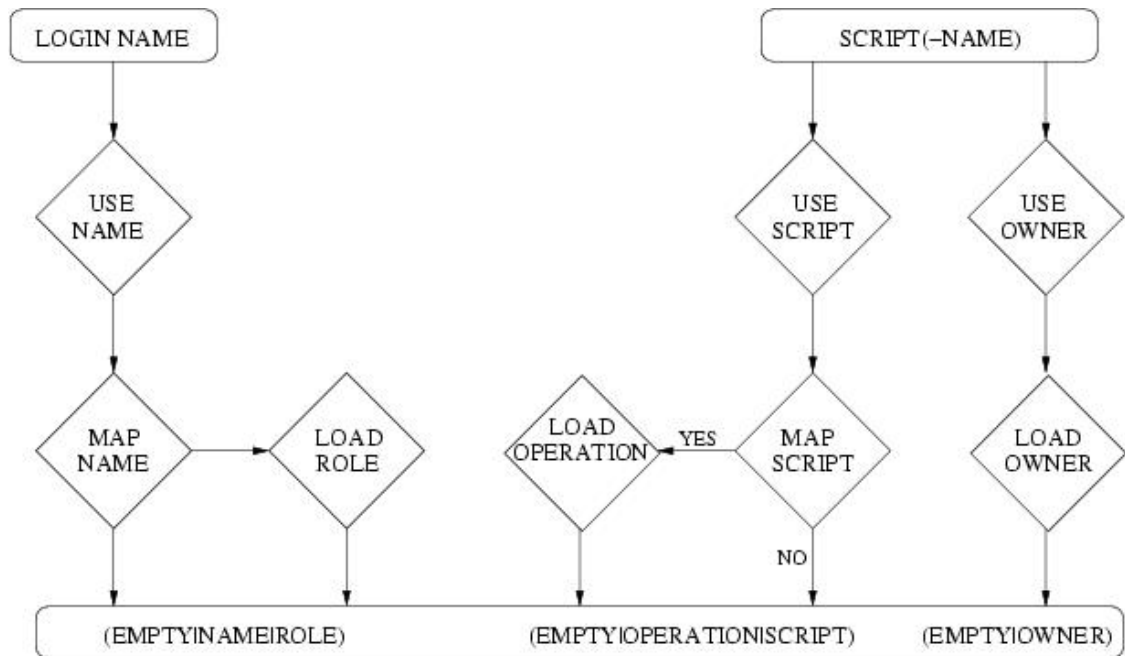


Figure 12.4. Access control list



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

Chapter 13. 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 slot-mechanism to support as many different logging technologies 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 will 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 be used 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 initialized 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 14. Webinterfaces

1. Interfacebuilding

OpenCA was designed for very fast and radical interface customization. Customization is understood 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 completely 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 selected 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 something 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 stylesheets 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:

Grammar 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 signinfo. 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 have to create a file `src/common/etc/rbac/cmds/filename.xml`. This file has to contain the configuration for the access control. Please add after this step default entry to the access control list in `src/common/etc/rbac/acl.xml`

If you want to add the new command directly in an existing installation then install the file in `lib/cmds` and create the access control file in `etc/rbac/cmds/`. Please don't forget to set a correct entry in the ACL at `etc/rbac/acl.xml`.

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, IS-
                   SUE_CERT_NEW, ISSUE_CERT_RENEW, ISSUE_CERT_PENDING, IS-
                   SUE_CERT_SIGNED,  ISSUE_CERT_APPROVED,  DELETE, DE-
                   LETE_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, RE-
                   VOKE_CERT_NEW, REVOKE_CERT_PENDING, REVOKE_CERT_SIGNED,
                   REVOKE_CERT_APPROVED,  DELETE,  DELETE_NEW,  DE-
                   LETE_PENDING, DELETE_SIGNED, DELETE_APPROVED
```

```
CmdRefs_viewCert INSTALL_CERT, LDAP, REVOCATION, SENDCERT, SEND_CERT_KEY,
                   VIEW_CSR, TOKENHANDLING, MAIL, SET_PUBLIC_PASSWD, DE-
                   LETE_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 re-configure 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 software requirements. 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 15. Hierarchy

1. Nodemanagement

2. Dataexchange

Chapter 16. 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. LDAP schema specification

1.1. 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

1.2. 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

1.3. 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 16.1. 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, opencaSCEP-Device
unstructuredAddress	cn	unstructuredName, unstructuredAddress, serialNumber, st, l, ou	top, device, opencaSCEP-Device
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)

LSC of the DN	filled attributes	filled attributes if present	objectclass stack
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.

1.4. 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 16.2. Schema usage for user certificates

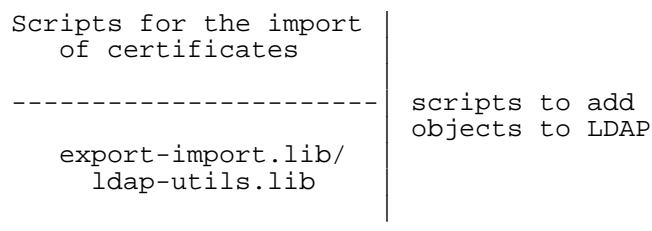
LSC of the DN	filled attributes	filled attributes if present	objectclass stack
cn	cn,sn	mail, o, st, l, ou	top, person, organizationalPerson, inetOrgPerson
sn	cn,sn	mail, o, st, l, ou	top, person, organizationalPerson, inetOrgPerson
emailAddress	cn,sn	mail, o, st, l, ou	top, person, organizationalPerson,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

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

2. Sourcecodeorganization

2.1. Structure of the code

Figure 16.1. LDAP source schema



OpenCA::LDAP

2.2. The relevant commands

- `addCertsLDAP` (puts all valid certs to LDAP)
- `addCrLDAP` (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`)

2.3. export-import.lib

`eximObjectToLDAP`

2.4. 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)

2.5. 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 17. Database

After version 0.9.2 the OpenCA core team decided to only support SQL databases because we need several features of SQL. Such aspects include functional requirements and performance issues. The result was a major change in the database backend because now all parts of the code can use such nice things like wildcards. We will discuss now the most important parts of the database design.

There are three major technologies which OpenCA needs:

1. Tables
2. Sequences
3. Indexes

Now let's start explaining all three issues.

1. Tables

The first technology is the most complex one. We have to handle two classes of tables - one for objects and one for hashes. OpenCA starts with using tables only to store objects. The interface was completely object oriented. It was only possible to handle cryptographic objects with the database backend (e.g. CRLs, CSRs, CRRs, certificates and CA certificates). Such an object will be stored as follows in the database:

serial	This is the serial number of the object.
data	This is the serialized object.
format	This field describes the format of the serialized object.
searchable attributes	The other columns of the database tables are reserved for the searchable attributes. OpenCA's database backend includes a specification for which stuff you can search in the object tables.

Hashes are handled a little bit different. You give the database backend a hash reference and you get a hash reference from the database backend. Such tables are used for content which is not object based. The following tables are hash based:

PRIVATE	This table is used to store any private data like keybackups, private keys and PINs. It only has three columns <code>private_key</code> , <code>data</code> and <code>format</code> where <code>format</code> is the type of the stored data.
STATEMACHINE	The statemachine has an own table to store all states of the active batch workflows. This is needed for performance reasons. If we would store the states of the statemachine in the data table then we have a big performance problem even if we use indexes. Otherwise it is easy with such a table to monitor active batch processors and to implement an error handling and escalation mechanism.
DATA	The table was originally developed to hold the data of the batch system but it can be used to store any other data too. The table has two keys a primary one and something like a global ID. The global ID is used to identify data from the same usecase. You can search on the table for one global key and you get all rows which

contain data of this context. This is today usually a batch workflow but it can be an isolated key backup or a user set of two private keys one for signing and one for encryption in the future.

The other columns are content type, array counter, number and string. The real names of the columns are of course different to avoid conflicts with the different SQL dialects. The design allows by this way the implementation of dynamic content detection and array support. More details about the usage of this table you can found at Chapter 18, *Batch System*.

2. Sequences

OpenCA uses sequences to get 100 percent safe serials. This is necessary for CSRs, CRRs, global IDs, private data and the data table. Additionally one sequence is planned for the auditing table. These sequences are accessible via `getNewSerial`. The major problem is the poor standardization of such sequence generators in SQL. Today every major database use it's own dialect and some vendors do not implement sequences until today.

3. Indexes

Indexes are really important because we need them for performance reasons especially for the batch system. Indexes are supported today by every database system. The only difficult thing is that some vendors implement them in a really table dependent way. If you cleanup your database please always remove the indexes first. If you remove the tables before the indexes then it can happen that you cannot remove the indexes from the database.

Chapter 18. 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 primarily 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 ::= oneline | multiline
```

```

oneline ::= [^\n]+
multiline ::= "\n-----BEGIN ".limiter."-----\n"
.content.\n-----END ".limiter."-----";
limiter ::= [A-Za-z0-9_-\.]

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 requires that you customize the batch functions. `__user_defined__` should be an uppercase word, there are no spaces allowed in the name. This identifier is used as filename, by default we put the file into the "data" directory of the process, if you want to put it somewhere else, you can give the directory after the name with the @ sign, e.g. "PIN@private" will create a file called PIN in the directory private. You can pass multiline data using mime-style limiters, the limiters will NOT go into the data file, so if you want to pass data which already has limiters and keep them, e.g. a PKCS7 structure, you have to add a second set of limiters. 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 18.1. batch_new_user.txt

```

jon_doe
jane_doe

```

Example 18.2. batch_new_process.txt

```

jon_doe
enc_key

jon_doe
sig_key

jane_doe
key_1

```

Example 18.3. 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
LOA_MODE IGNORE

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
LOA_MODE USE_IT
LOA 20
importedPIN@private
-----BEGIN MYPIN-----
-----BEGIN PKCS7-----
SmXGmDTsQXiRmOvuWWRIGVz3ZjVGRKZdJBGGJf1DjkYYKQFbYSHA9pY9BYIeZrp8
.....some more lines.....
7fo=
-----END PKCS7-----
-----END MYPIN-----

```

The above example describes the classic import mechanism with three files. So you can create user, process and data in three single steps, and if you see something going wrong (e.g. creating a user already there) you can stop the process. OpenCA provides a QuickImport Feature that needs only the last file (batch_process_data.txt) and creates user, process and process data in a single step. It will take care and not override already existing process, but create more than one process even if the user exists in the system. Nevertheless the QuickImport Feature should fit 98% users needs.

4. Database background

To understand our approach fully it is now a good time to explain the database model in the background. OpenCA includes one special table which we use for data integration - the table data. All other tables like statemachine, request, certificate or private can only be used to store one special type of data in it but how do you want put all these data objects together?

The idea was a table which includes a second key to identify all rows which are from the same data source (sometimes called user or process). This second key is called global ID at OpenCA. All user

defined variables of the batch system are called *BATCH_**. References to other tables have names like **_REF*. Today there are the following references:

- *BATCH_PIN_REF*
- *KEY_REF*
- *KEYBACKUP_REF*
- *CSR_REF*
- *CERT_REF*

The batch system uses the table private really heavy. PINs, private keys and keybackups are stored in this table. The private data is accessible for the batch system via the references in the table data.

The primary key of the table statemachine is the global ID. The states were only separated from the table data because of performance issues. It is also a good idea to have a separate table if you have a separate functional component like the batch system.

5. 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.

6. Default workflow

Table 18.1. 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	backupid_key	
check_csr_params	backupid_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	
enroll_pin	new_cert	enrolled_pin	
enroll_pkcs12	enrolled_pin	enrolled_pkcs12	

7. 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 Chapter 11, *Cryptolayer*. 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 always 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 crashes.

8. Performance

8.1. 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 18.2. 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 19. Packaging

1. Common Notices

1.1. Required Perl modules

- XML::Twig
- MIME::Tools
- Net::LDAP - perl-ldap
- CGI
- CGI::Session
- 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

Optionally it is possible to set a RPM specific package number to all packages generated (the default is 0). To change the release number for the generated packages change the `RELEASE` variable in `suse/Makefile` to the desired value and run **make update-spec-release** before starting the package build. This command locally changes the 'Release:' tags of all spec files for the packaging process.

Example 19.1. 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. Dependency 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 20. 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 not 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 world-wide 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 month and 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 SSL-Leay 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. References

Many people ask who uses OpenCA. The problem is that we are a non-profit organization and so we have real problems to get references of existing systems. If you, your organization or your company use OpenCA and have no problem to publish this fact then please mail us. This helps us to give others an overview and they can ask a local user for it's experiences. This is sometimes more important than any other fact. We can add a small description too.

1. Universities

AU - KBC Research Centre
[<http://www.au-kbc.org>]

We use OpenCA in our lab & have delivered the same to our main university for their internal use. Infact we started giving commercial support to some of our local customers too.
MIT [<http://www.mitindia.edu>] (Madras Institute of Technology),
Anna University Campus [<http://www.annauniv.edu>]
Chennai - 44, TN, India
Contact: <mmurali@au-kbc.org>

Humboldt-University of Berlin,
Computing Centre

We use three OpenCA PKIs. Two Sub-CAs [<https://ra.hu-berlin.de>] in the hierarchy of the DFN-PCA [<http://www.dfn-pca.de>]. This is the Root-CA of Germany's National Research and Education Network [<http://www.dfn.de>]. One CA is a private isolated root CA which is based on OpenCA 0.9.1. We use this for internal communication including key recovery. You can contact us via <pki@hu-berlin.de>.

Technische Universitaet Munich,
Department of Electrical Engineering
and Information Technology

We are running one OpenCA since 2003 and use it for secure eMail for our students and employees (approx. 2500). Currently we set up a new structure based on the 0.9.2 release with one Root-CA and two Sub-CAs for eMail security and keybased shell access. We plan to certify the system within the hierarchy of the DFN-PCA [<http://www.dfn-pca.de>] - the PKI of Germany's National Research and Education Network. See <http://www.ldv.ei.tum.de> for details - questions are welcome.

Appendix C. Internationalization - i18n

OpenCA supports internationalization (i18n) since version 0.9.1. Full i18n support is available with version 0.9.2. There are several different translation teams. Following you can find the available translation and the contact informations. Please notice that we usually don't publish private addresses here to reduce the spam of the authors. If there are real problems then we have the emailaddresses of course and we will provide you with them. If you want to create a new translation feel free to contact us on one of our mailing lists.

1. de_DE

The German translation is managed by Michael Bell from Humboldt-University of Berlin but there are more than one German OpenCA developer. So the easiest way is to write a mail to `openca-users@lists.sf.net` if you have a problem with the translation.

2. it_IT

Massimiliano "madwolf" Pala is the actual manager of the Italian translation. He is OpenCA's project manager and the original founder of the project. You can contact him via our mailing lists too.

3. ja_JP

The Japanese translation was created by Takahiro Tsuji. You can conact him for any problems at `<openca@poplar.complex.eng.hokudai.ac.jp>`. He works at the ITS (Information Technology and Systems) laboratory of Hokkaido University in Japan. It was the first translation which don't based on conventional characters.

4. pl_PL

Franciszek "Franz" Lewenda from the Research and Academic Computer Network (NASK) creates the Polish translation. You can reach him via our mailing lists. This was the first non-iso-8859-1 translation.

5. sl_SI

The Slovene translation was created and is for now managed by Janez Pirc. He works for SETCCE (Security Technology Competence Centre, <http://www.setcce.org>) which uses OpenCA to manage their own Certificate Authority under the name SI-CA (<http://www.si-ca.org>). This was the first UTF-8 translation of OpenCA. If you want to contact the translator, just send an e-mail to `openca[at]si-ca[dot]org`.

Appendix D. Authors and Contributors

Here you can find all the authors and contributors to this guide.

1. Martin Bartosch

nCipher, Oracle, CA Rollover documentation.

2. Michael Bell

Configuration issues, batch system, LDAP, FAQ maintenance and all content which is not explicitly written by others :)

3. Chris Covell

LDAP stuff and Web interface documentation.

4. Massimiliano Pala

History.

5. Ulrich Bathels

Section 11.1, “Sendmail with basic SMTP authentication”

6. Ashutosh Jaiswal

Proofreading and content editing of the documentation.

7. FAQ

Several people have helped us on the mailing lists. We have extracted many hints and workarounds from these lists and have added them to the FAQ. We try to list here the guys that have helped us. If we have forgotten somebody please help us and send a notice.

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

Appendix E. 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 guaranteed 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 approve 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 approved 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 must 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
- key sizes are now choosable for IE users too
- much better CSR editing

- additional attributes for requests (e.g. telephonenumber)
- 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` until version 1.7. We strongly recommend that you update to a newer version. Some workarounds are described at Section 2.2.2, "Signing Data".

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 `ca.cert.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 error message from OpenSSL. If the error message 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 Kaspersky. Like often with Microsoft programs it is not clear why Outlook crashes 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 E.1. General error 6751 during certificate issuing

```
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 errcode 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 E.2. Bad passphrase error log during certificate issuing

```
[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_REQUE
[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_REQUE
[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_REQUE
```

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.

2.14. What does I have to do if I create a new release?

This defines all necessary steps for a new release and is mandatory for release candidates too. Steps which are on mandatory for normal releases or release candidates are marked.

1. Go to CVS module directory `openca-0.9`
2. Edit `Makefile.devel` and fix the minor release

3. `Commit Makefile.devel`
4. `cd ..`
5. `cvs tag -R openca_V_E_R_S_I_O_N openca-0.9`
6. `cd openca-0.9`
7. `make -f Makefile.devel dist`
8. `scp openca-0.9.2*.tar.gz username@ftp.openca.org:ftp/releases/`
9. `ftp upload.sf.net`
10. Login: **anonymous**
11. Passwd: **your emailaddress**
12. `cd incoming`
13. `put openca-0.9.2*.tar.gz`
14. Go to sourceforge.net and release the file for project openca
15. Add a release for OpenCA at freshmeat.net
16. Add news message to news area of OpenCA.org
17. Send a mail to openca-users, openca-devel, openca-announce

2.15. How can I configure Mozilla for OCSP?

Which string should be filled in the `Service URL` field of the Mozilla Preferences/Validation assuming that `10.13.1.13` is my CRL IP?

Well, it depends on your configuration (check the `ocspd.conf`). Anyway by default you should use `http://10.13.1.13:2560/`.

2.16. Error 7211021: Cannot create request!

Sometimes you get the following error message.

Example E.3. Error 7211021: Cannot create request!

```
Error 7211021
General Error. Cannot create request!

(OpenCA::REQ->new: Cannot create new request.
Backend fails with errorcode 7712071.
OpenCA::OpenSSL->genReq: Cannot execute command (7777067).
problems making Certificate Request 24649:
error:0D07A097:
asn1 encoding routines:
ASN1_mbstring_copy:
string too long:
```

```
    a_mbstr.c:154:maxsize=2
error in req
).
```

The reason is very simple you entered more than two characters for the ISO country code. Please check you form and the configuration for the used country code. All ISO country codes are two characters long - not one character and not more than two characters.

3. Installation Issues

3.1. FreeBSD, OpenBSD and OpenCA

Please ignore this section for OpenCA 0.9.2.0+ and snapshots after 2004-Sep-15.

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. Solaris and OpenCA

Please ignore this section for OpenCA 0.9.2.0+ and snapshots after 2004-Sep-15.

3.2.1. make

Solaris has the same problems like the BSDs. The Solaris make doesn't work with OpenCA's makefiles. The solution is to install the binutils and gmake from sunsolve. You can configure OpenCA after the installation at 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.

3.3. What is a hierarchy level?

OpenCA 0.9.1 and earlier 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 `OPEN-CADIR/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 choose one of the above defined options and then you have to edit the options for the dataexchange manually like described in the administrator guide.

3.4. Undefined subroutine `&main::xyz`

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

Example E.4. 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.

3.5. Symbolic link installation failed

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

Example E.5. 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.

3.6. 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.

3.7. Conflicting Modules

Sometimes there are reports about error messages 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 are two well known issues during the startup - wrong permissions and duplicate installations of a module. Please always 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 Perl module `XML::Twig`. This can cause a crash of the daemon at startup or always empty results for a configuration file. First you should search for the different version of

XML::Twig.

Example E.6. 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.

3.8. 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 error message 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 interpret our configuration files.

3.9. Unknown Login Type

Sometimes you get the error message 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 know 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.

3.10. 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 E.7. Type Mismatch on Internet Explorer

```
Line: 88
Char: 1
Error: Type Mismatch: 'GenReq'
Code: 0
URL://xxxxxxxxxxxxxxxxxxxx/cgi-bin/pub/pki?cmd=basic_csr
```

3.11. openca(_rc) start failed

Example E.8. Failed startup with wrong Net::Server version

```
#> /etc/rc.d/init.d/openca start
Starting OpenCA: Process Backgrounded
2004/04/22-12:34:19 OpenCA::Server (type Net::Server::Fork) starting!
pid(3195)
Binding to Unix socket file /var/lib/openca/tmp/openca_socket using
SOCK_STREAM
Setting gid to "73 73"
Setting uid to "73"
Couldn't POSIX::setuid to "73" []
```

This error message is caused by an outdated perl module Net::Server. Please update to a Net::Server version higher or equal to 0.86. The server itself will usually start nevertheless the error is reported or not. Please use **ps -efa** to check for the two necessary daemons - XML cache and OpenCA server.

3.12. Missing modules

3.12.1. XML::Parser

If you see one of the following error messages during the compilation or installation of OpenCA then you are missing the perl module XML::Parser.

The first symptom can be a failing make in XML::Twig. Please notice that a second make in the same directory does not show any anomaly. Sometimes only the first make shows up an error message because of the missing XML parser.

Example E.9. failing XML parsing during configuration

```

Checking if your kit is complete...
Looks good
Warning: prerequisite XML::Parser failed to load: Can't locate
XML/Parser.pm in @INC (@INC contains: /usr/local/lib/perl/5.6.1
/usr/local/share/perl/5.6.1 /usr/lib/perl5 /usr/share/perl5
/usr/lib/perl/5.6.1 /usr/share/perl/5.6.1 /usr/local/lib/site_perl .) at
(eval 4) line 3.
Writing Makefile for XML::Twig
make[4]: Leaving directory `/home/ca/openca-0.9.2-RC4/src/modules'
make[4]: Entering directory
`/home/ca/openca-0.9.2-RC4/src/modules/XML-Twig-3.09'
/usr/bin/perl speedup Twig.pm.slow > Twig.pm
Can't locate XML/Parser.pm in @INC (@INC contains:
/usr/local/lib/perl/5.6.1 /usr/local/share/perl/5.6.1 /usr/lib/perl5
/usr/share/perl5 /usr/lib/perl/5.6.1 /usr/share/perl/5.6.1
/usr/local/lib/site_perl .) at speedup line 5.
BEGIN failed--compilation aborted at speedup line 5.
make[4]: *** [Twig.pm] Error 2
make[4]: Leaving directory
`/home/ca/openca-0.9.2-RC4/src/modules/XML-Twig-3.09'

```

The second symptom can happen if you try to configure your OpenCA software.

Example E.10. failing XML parsing during configuration

```

> ./configure_etc.sh
> =====
> fixing directory: /home/ca//OpenCA/etc
> =====
> -----begin file-----
> template: ./access_control/ca.xml.template
> target: ./access_control/ca.xml
> XML/Twig.pm did not return a true value at
> /usr/local/share/perl/5.6.1/OpenCA/Tools.pm line 351.
> BEGIN failed--compilation aborted at
> /usr/local/share/perl/5.6.1/OpenCA/Tools.pm line 351.
> Compilation failed in require at /home/ca//bin/openca-configure line 7.
> BEGIN failed--compilation aborted at /home/ca//bin/openca-configure line 7.

```

If you try to locate the installed Twig.pm files and check there length then usually only Twig.pm.slow is not empty.

Example E.11. empty Twig.pm files because of missing XML::Parser

```
> > locate Twig.pm
> /home/ca/openca-0.9.2-RC4/src/modules/XML-Twig-3.09/blib/lib/XML/Twig.pm
> /home/ca/openca-0.9.2-RC4/src/modules/XML-Twig-3.09/Twig.pm
> /home/ca/openca-0.9.2-RC4/src/modules/XML-Twig-3.09/Twig.pm.slow
> /usr/local/share/perl/5.6.1/XML/Twig.pm
```

3.13. Trouble with databases, database drivers and Perl DBI

Sometimes it happens that the database driver or Perl DBI is too old to work properly. This can happen especially for CRLs. You can see the following message type in the error log:

Example E.12. Too old DBD::Pg or DBI trouble

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 E.13. 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 E.14. `./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 E.15. 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. Please note, that for Apache 2.0.* you also need to add in the line "SSLOptions +OptRenegotiate" in order for Apache to re-negotiate the access permissions correctly.

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 E.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 E.16. OCSF 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 E.17. OCSP configuration for http

```
[ ocspl_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 error message 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 hundreds 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. 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/UnidibleICA4/etc
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>
```

4.15. 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.

4.16. 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 existing 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.

4.17. Unknown login type

If you get an error message which indicates that there was an unknown login configured then please see Section 3.7, "Conflicting Modules".

4.18. Cannot initialize cryptoshell but OpenSSL path is correct

If you get an error message that OpenCA cannot initialize the cryptoshell and after the startup no daemon is up - whether the XML cache nor the OpenCA daemon then you have perhaps uncleanly stopped OpenCA's daemons. Please check `OPENCADIR/var/tmp/` for any socket files. If OpenCA was killed during shutdown without an explicit shutdown by `openca_rc` or `openca_stop` then the still existing socketfiles can block the startup of the daemons. This behaviour is a must to avoid implicit killed daemons if you try to start your daemon twice.

4.19. Emailaddress in subjectAltName but not in CA subject

How can one create a CA certificate so that the DN does not contain the email address, but the alternate name does? You can configure this in `OPENCADIR/etc/openssl/openssl.cnf`. Set the subject alternative name for the `v3_ca` to the required emailaddress. Don't add the emailaddress to the subject of

the new request.

Example E.18. emailaddress for subjectAltName in CA certs

```
subjectAltName=email:xyz@abc.org
```

4.20. Missing environment variables from SSL

Sometimes users see the following error message:

Example E.19. Missing mod_ssl standard environment variables

```
General Error 6251043:  
Aborting connection - you are using a too short symmetric keylength ().
```

This means originally that you are using a symmetric key which is shorter than specified in `OPEN-CADIR/etc/access_control/ra.xml`. Usually the symmetric cipher must have a length greater or equal 128. If you are using a Mozilla browser then you can click on the small lock to get informations about the used session cipher.

The empty brackets at the end of the error message means that there is no keylength specified. This is a perfect indication of a problem with the environment variables. Usually the standard environment variables for SSL are not activated in your Apache. You can configure it like follows:

Example E.20. SSL environment variable configuration for Apache

```
SSLOptions +StdEnvVars +ExportCertData
```

Please NEVER set the required symmetric keylength to 0. This is security hole.

4.21. Problems with the country name during PKCS#10 requests

Sometimes there are problems with PKCS#10 requests. OpenCA displays an error messages like this

one:

Example E.21.

```
Error 700
General Error. Your request has to include C=.."
```

Please check config.xml. Usually you forgot to configure the country name. If you don't want to run **configure_etc.sh** again then please check the files in `OPENCADIR/etc/servers/` for the configuration of the country code.

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 starts 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 in-

stalled 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 E.22. 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 language?

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
-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.

8.4. MySQL and SET NAMES errormessages

Sometimes you find errors related to **SET NAMES** in the logs of your SQL server if you use MySQL before version 4.1. This happens because MySQL supports this part of SQL only since version 4.1. You can ignore these errormessages.

Databases and their tables are usually created for a specific language. OpenCA is able to support all available languages at all times since version 0.9.2. This requires the support for several different character encodings like UTF-8, ISO 8859-1, ISO 8859-2 and ISO 8859-15. Therefore OpenCA must set the used character encoding before it uses a database. If we don't do this then the database can abort actions because of failing transformations. SQL defines a syntax for such operation which include **SET NAMES**. MySQL implemented this only since 4.1 and therefore we run the SQL command in a way which don't abort transactions. Therefore you can ignore errormessages from your MySQL database if the version is smaller than 4.1.

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. Zuercher Hochschule Winterthur. July 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 Mueller, and With contributions from Bob Stayton. O'Reilly & Associates. 31st December 2003.
- The DocBook Wiki* [<http://docbook.org/wiki/moin.cgi/>]. DocBook.
- SecClab's webpage* [<http://secclab.mozdev.org/>]. Luis Fernando Pardo.
- AutoSSCEP's webpage* [<http://autosscep.spe.net/>]. Paolo Prandini.
- OpenTSA's webpage* [<http://www.opentsa.org/>]. OpenTSA project.
- OpenCA Live CD* [<http://www.dartmouth.edu/%7Edeployki/CA/InstallOpenCALiveCD.html>]. Dartmouth PKI Lab Outreach. Kevin Mitcham.

Glossary

ADS	Active Directory Services
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
GUID	The Global Unique Identifier is a 16 byte string for an object the ADS. Every domain controller have such a GUID for example and they must be present in the certificates of these domain controllers.
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 network components 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 mail servers like sendmail to exchange the mails. The protocol is used for mail transfer from simple MTAs like Mozilla to servers like sendmail and for transfers from server to server.
SMTP over TLS	is nothing else than SMTP through a TLS tunnel. It protects the communication between a mail server and a mail user agent if the user reads and manages its mail.
SPKAC	Signed Public Key And Challenge is a standard for CSRs from Netscape.
SSL	Secure Socket Layer is a transport layer security protocol. It was one of the first certificate based security protocols for tunneling. Netscape developed this protocol for its web browser Navigator. TLS is the standardized successor 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 too fast ;-)
TLS	Transport Layer Security is a transport 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.
TTP	Trusted Third Party is usually a trustworthy external CA.
UPN	The Universal Principle Name is the user account plus the domain name. They are connected with an at-sign @. Example: john_doe@company.com.

Appendix F. Strategy

This appendix gives an overview of the current strategic direction of the OpenCA development. As all things it is subject to change, but this is the current thinking.

1. The Strategy Behind OpenCA Development

1.1. Scalability

A statement from the OpenCA team to say that for a given server and database environment what is the expected volume of certificates. This is important for users planning installations.

1.2. Command Line API to CA and RA Functions

This would allow for complex scripts to be written around the OpenCA environment, hopefully paving the way for future modifications. This would also lead to the possibility of XKMS, CMS, SOAP based clients.

This would include a fully documented PERL API scripting interface.

1.3. Automation functions

Automation of regular operations like: CRL production, certificate signing. This is important in production environments where you do not want Operations staff to have to manually produce regular CRLs, etc.

1.4. On-line CA model option

To accommodate an on-line CA model. i.e. a user can request a certificate and in the same session get the requested cert back. This can be used for "free email certs" or in closed user groups where only certain people have access to the public interface. It may be that this would only work with CA root key in hardware, or a special CA user logged on on a secure terminal to give the environment access to the CA password. In addition to this, the CA may keep a log of the number of times it was accessed.

1.5. High Risk Environment Mode

A high risk environment mode, which is based on a cd-rom or some similar write protected media, changeable configuration data and exchange data are hold on writeable media (like usb-based-hardware, maybe encryptable), and the os supports something like se-linux or similar.

1.6. Audit logging

Audit of RA and CA operations to a tamper proof signed log. This is possibly a requirement to achieve any form of accreditation.

1.7. Script/environment validation

A function that ensures OpenCA is running in a "known" environment. Perhaps md5 signature creation (after installation) and run time validation.

1.8. Automated CA rollover

When reaching the end of the CA certificate lifetime, there is a certain point in time after which no usable end entity certificates can be issued whose desired validity fully fits into the CA certificates validity.

To address this problem an automated CA Rollover is implemented. The basic idea is to have multiple issuing CAs (with overlapping certificate validity) that are logically responsible for the same set of certificates. OpenCA will automatically detect which CA to use for a given operation.

1.8.1. External and Internal CAs

An *OpenCA installation* consists of the program installation, including the binaries, web frontends, configuration files, etc.

An OpenCA installation may provide one or more externally visible *CA instances* or *External CAs*. A unique *External CA Identifier* is assigned to each External CA that is used to distinguish between the individual instances. Each External CA provides its own independent name space in terms of profile, common name, serial number, access control etc. This makes it possible to run different and completely independent CAs in one single installation without having to install multiple program installations. Selection between instances for end users and administrator staff is delegated to either the web server running the CA installation (e. g. by using different URLs or port numbers for instance distinction) or may be performed by some kind of selection method on the CA login screen. An External CA (or CA instance) encompasses a complete OpenCA configuration set.

Each External CA may consist of an arbitrary number (zero or more) of *Internal CAs*. All these Internal CAs should be capable of issuing certificates for the namespace defined by the External CA. Internal CAs will usually use the same Certificate Profile and very similar configuration, but this is neither mandatory nor technically enforced by the OpenCA framework. In particular, each Internal CA uses its own CA certificate and private key. Each Internal CA is given a unique *internal identifier* that is used internally by OpenCA to distinguish between internal CAs. It is not allowed to change the internal identifier after it has been assigned to the Internal CA.

For convenience it is possible to assign *symbolic names* to each Internal or External CA. The symbolic name is only used for display purposes and may be changed at any time by the administrators. All references (database, configuration) to External and Internal CAs are only done by using the external and internal CA identifiers.

On startup the OpenCA system examines all External CAs that are configured. For each External CA OpenCA determines which Internal CAs belong to this CA instance and analyzes the corresponding Internal CA certificate. The validity information of each CA certificate is stored internally for later use by the CA request dispatcher.

In order to make CA rollover work, administrators must make sure that an Internal CA exists that is capable of taking over the certificate issuance duties of the expiring Internal CA. If no Rollover CA exists, the CA system will not be able to find a suitable candidate Internal CA for the requested operation and will stop working after the last Internal CA exceeds a certain point in time after which it is not possible to issue end entity certificates with the required validity.

1.8.2. Request processing

In the following sections a request is any operation that must be performed by a CA, e. g. CRL generation, certificate issuance, certificate revocation or certificate renewal.

Whenever a CA operation is requested the OpenCA system first determines the External CA the request belongs to. The request should contain an indication of the External CA it should be applied to, e. g. when originating from a web frontend the CGI script will include information about the originating CA Instance in the request. This indication must be the External CA Identifier as defined above.

In some cases it is possible that requests arrive at the system without an explicit indication of the responsible External CA. To handle such cases it should be possible to configure a single External CA that should act as the default CA. If no default defined, incoming requests without an explicit reference to an External CA must be discarded. An error should be delivered to the requester if possible.

Note

Although a request may contain an External CA Identifier that makes it possible to select the responsible External CA for the request, the request cannot and must not contain an Internal CA Identifier. Consequently the responsible Internal CA is always automatically determined by the OpenCA system.

1.8.3. Dispatching requests to Internal CAs

After the OpenCA system has identified the responsible External CA for a request it must determine the Internal CA that should process the request. All valid Internal CAs that are configured for an External CA are possible candidates for processing the request (unless they are explicitly disabled in the configuration).

1.8.3.1. Rollover requirement 1: automatic consideration of CA validity

Requests will only be processed by an Internal CA if this Internal CA is suitable for the requested operation. This means that an Internal CA whose validity is not a superset of the requested end entity certificate validity is not considered for request processing at all.

1.8.3.2. Rollover requirement 2: request dispatch decision

After determining the Internal CAs that are candidates for request processing, the system tries to identify which of the remaining CAs should actually process the request.

The handling is differs depending on the individual request type:

1.8.3.2.1. CRL issuance

After a request for CRL issuance has been received for an External CA, the request is dispatched to all configured and operational Internal CAs for this External CA. This means that all CAs that are available should issue CRLs.

1.8.3.2.2. Certificate issuance

In order to determine the responsible Internal CA for a given certificate request the following algorithm is used:

- Determine requested end entity certificate validity (NotBefore and NotAfter dates)
- From all possible Internal CAs determine the ones whose CA certificate validity allow to issue the requested end entity certificate
- From the remaining Internal CAs determine the one with highest NotBefore date
- Dispatch the request to the Internal CA that was determined. Report an error if no possible CA candidate was found.

1.8.3.2.3. Certificate revocation

Determine the Internal CA that issued the certificate that is to be revoked. Dispatch the revocation request to this Internal CA.

1.8.3.2.4. Certificate renewal

The following section describes a proposal for certificate renewal (reissuance of a certificate for an end-entity that has been granted a certificate before). The actual decision if a certificate may be renewed depends on the policy for the External CA and should be configurable.

The following algorithm is proposed for certificate renewal:

- Reject request if no previous certificate exists for the requested Common Name.
- Reject request if already more than one valid certificate exists for the requested Common Name (allow a maximum number of two valid certificates for one single CN).
- Determine the NotBefore and NotAfter date of the requested end entity certificate (proposed method: use NotAfter of existing certificate as NotBefore of renewal certificate. To determine the Not-After date of the renewal certificate add default end entity validity period to NotAfter date of existing certificate)
- Proceed with the algorithm outlined in "Certificate Issuance"

1.9. Function to process signing and encryption keys in one go

OpenCA could introduce the idea of "certificate profiles" where a user "requests" once but gets a "Profile" of certificate types. Secure storage and recovery of encryption keys would be part of this mechanism. The start of this is in the new Batch processes in the form of the "Process".

1.10. Secure storage and recovery of encryption keys

A function to provide (optional) key backup and recovery.

1.11. Web based OpenCA configuration and management

Enhancing the existing management screens to allow management of certificate roles and extensions, access control settings and node management i.e. a front end to the OpenSSL config files.

1.12. Improved key lifecycle management

Screens to allow users to renew their certificates, modify DN's etc.

1.13. Authentication via a third party

The ability to allow a user to request a certificate and authenticate themselves the authentication token is then checked against an independent directory.

1.14. Improved debugging support

Sometimes it is difficult to figure out which functions are called by the system. Currently debugging is enabled in various configuration sections.

1.15. Improved error handling

Sometimes OpenCA reports crude error messages on seemingly harmless error conditions. When checking the code it was often something like an uninitialized variable that was used to call a method on.

1.16. Accreditation

Achieve Common Criteria/FIPS accreditation ! This is a long way off, but with OpenSSL being pushed through, then it may be possible !!!