Table of contents

Chapter 1
Administratorguide

1.1 Installation

The installation of OpenCA is based on autoconf and automake the wellknown GNU-tools. The basic steps are simple.

1. Download a snapshot from OpenCA.

2. Unzip the snapshot (gunzip filename.tar.gz or gzip -u filename.tar.gz).
3. Untar the snapshot (tar -xvf filename.tar).
4. Change into the directory of the snapshot.
5. Configure the sources (see 1.1.2).
6. Run make (see).
7. Install the software (see).

The first four steps are quite simple. The problems start with the configuration of the software but before you start reading about the problems please read the requirements carefully because a lot of problems are caused by wrong versions or missing modules.

1.1.1 Requirements

- Perl 5.6.1 or higher

  You can enforce the usage of Perl 5.6.1 during configure if you are using more than one version of Perl on you system. This can happen if the vendor of your operating system ships the system with a very old version of Perl. You must simply call configure in the following way

  env PERL=/path/to/perl ./configure ...
- OpenSSL 0.9.7 20020415 or newer

1.1.2 Configure the sources

There are four big parts in the configuration of the software.

1. placement of the normal OpenCA-software
2. placement of Perlmodules
3. placement of Apache related stuff
4. Preconfiguration of the software

1.1.2.1 Placement of the normal OpenCA-software

Before we start to explain the different options you must understand the general structure. We know a maindirectory which is usually /usr/local/OpenCA. The option to define the directory is —with-openca-prefix. This directory contains several other directories:

- etc
- lib
- var

The names have the meanings like in several other Open Source packages. etc contains the configuration, lib contains the softwareparts which are no binaries and var contains all dynamic data. The binaries (the only exception) are placed by default in /usr/local/bin. You can modify this behaviour by setting –prefix or –exec-prefix. The full structure looks like follows:

```
etc/
etc/openssl/
etc/openssl/openssl/
etc/openssl/extfiles/
etc/rbac/
etc/rbac/modules/
etc/rbac/operations/
etc/rbac/rights/
```

```
etc/rbac/roles/
etc/rbac/scripts/
etc/servers/
lib/
lib/cmds/
lib/functions/
lib/servers/
var/
var/crypto/
var/crypto/cacerts/
var/crypto/certs/
var/crypto/chain/
var/crypto/crls/
var/crypto/keys/
var/crypto/reqs/
var/db/
var/mail/
var/mail/crins/
var/tmp/
```

The most important directories for non-developers are etc/servers/ and etc/openssl/.

1.1.2.2 Placement of Perlmodules

The perl modules can be placed in a special directory to remove them easily from your machine after testing OpenCA. The option is –with-module-prefix. We strongly recommend you to set this option to not mix your normal system with OpenCA.

1.1.2.3 Placement of Apache related stuff

We know two different groups of users until today. The users which install OpenCA on a Linux with an Apache with cgi-bin- and htdocs-directory somewhere in the system but with OpenCA on top of the Apache. The other users are the real Apache-admins which compile their Apache by hand and want to embed OpenCA somewhere in the system. We allow the following options to give you an easy but flexible way of installing the software:

General Options

–with-web-host=WEBHOST
: sets the host for all services like www, ldap, ocsp, scep

–with-httpd-host=ARG
: sets the hopstname of the webserver which hosts the OpenCA-software

–with-httpd-user=ARG
: sets the username which uses the Apache. This is needed to set the ownership of the files correctly which must be writable by the software.

–with-httpd-group=ARG
: sets the group which uses the Apache

–with-httpd-fs-prefix=HTTPDFSPREIFX
: sets httpd filesystem prefix (default is PREFIX/apache)

–with-httpd-url-prefix=HTTPDURLPREFIX
: sets httpd URL prefix (default is empty which means OpenCA will be installed in the servers document root)

–with-htdocs-fs-prefix=HTDOCSFSPREFIX
: sets htdocs filesystem prefix (default is HTTPDFSPREFIX/htdocs)

–with-htdocs-url-prefix=HTDOCSURLPREFIX
: sets htdocs URL prefix (default is HTTPDURLPREFIX)

–with-cgi-fs-prefix=CGIFSPREFIX
sets CGI filesystem prefix (default is HTTPDFSPREFIX/cgi-bin)

–with-cgi-url-prefix=CGIURLPREFIX
sets CGI URL prefix (default is HTTPDURLPREFIX/cgi-bin)

Certification Authority

–with-ca-htdocs-fs-prefix=DIR
sets CA htdocs directory (default is HTDOCSFSPREFIX/ca)

–with-ca-htdocs-url-prefix=DIR
sets CA htdocs URL-prefix (default is HTDOCSURLPREFIX/ca)

–with-ca-cgi-fs-prefix=DIR
sets CA cgi directory (default is CGIFSPREFIX/ca)

–with-ca-cgi-url-prefix=DIR
sets CA cgi URL-prefix (default is CGIURLPREFIX/ca)

Administration of the Online Components

–with-online-htdocs-fs-prefix=DIR
sets Admin htdocs directory (default is HTDOCSFSPREFIX/online)

–with-online-htdocs-url-prefix=DIR
sets Admin htdocs URL-prefix (default is HTDOCSURLPREFIX/online)

–with-online-cgi-fs-prefix=DIR
sets Anline cgi directory (default is CGIFSPREFIX/online)

–with-online-cgi-url-prefix=DIR
sets Admin cgi URL-prefix (default is CGIURLPREFIX/online)

Registration Authority

–with-ra-htdocs-fs-prefix=DIR
sets RA htdocs directory (default is HTDOCSFSPREFIX/ra)

–with-ra-htdocs-url-prefix=DIR
sets RA htdocs URL-prefix (default is HTDOCSURLPREFIX/ra)

–with-ra-cgi-fs-prefix=DIR
sets RA cgi directory (default is CGIFSPREFIX/ra)

–with-ra-cgi-url-prefix=DIR
sets RA cgi URL-prefix (default is CGIURLPREFIX/ra)

LDAP Gateway

–with-ldap-htdocs-fs-prefix=DIR
sets LDAP htdocs directory (default is HTDOCSFSPREFIX/ldap)

–with-ldap-htdocs-url-prefix=DIR
sets LDAP htdocs URL-prefix (default is HTDOCSURLPREFIX/ldpap)

–with-ldap-cgi-fs-prefix=DIR
sets LDAP cgi directory (default is CGIFSPREFIX/ldap)

–with-ldap-cgi-url-prefix=DIR
sets LDAP cgi URL-prefix (default is CGIURLPREFIX/ldap)

Public Gateway

–with-pub-htdocs-fs-prefix=DIR
sets public htdocs directory (default is HTDOCSFSPREFIX/pub)

–with-pub-htdocs-url-prefix=DIR
sets public htdocs URL-prefix (default is HTDOCSURLPREFIX/pub)

–with-pub-cgi-fs-prefix=DIR

sets public cgi directory (default is CGIFSPREFIX/pub)

–with-pub-cgi-url-prefix=DIR
sets public cgi URL-prefix (default is CGIURLPREFIX/pub)

Like you can see it is easy to configure OpenCA for testing only but is also possible to integrate OpenCA into a complex webserver.

1.1.2.4 Preconfiguration of software

All the options which were not described above are used to configure the software itself. This include things like databases, LDAP-servers and names of the installed components (e.g. –with-ra-prefix is the name of the directory with all special softwarecomponents of the Registration Authority in lib/servers/ and the name of the configuration file in etc/servers/).

The details are not only necessary for the software itself. Some parameters are used for the configuration of the certificates. So please be careful if you setup a real trustcenter.

1.2 Configuration

This chapter describes how you can configure OpenCA.

1.2.1 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.

1.2.1.1 Servercertificates

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. Thi 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 in the Netscape defines. 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 comliant but supports the cruel behaviour of Netscape.

1.2.1.2 Troubleshooting

All cryptographic opertions fails because the software cannot find openssl.

Please check the configured path to the binary of OpenSSL in etc/servers/*.conf. The variable which is the problem is OPENSSL.

Apache's error.log reports a nonexistent option "-subj" of openssl req.

You must install and use OpenSSL v0.9.7 or later. This is necessary because some options are only supported in the later versions. OpenSSL v0.9.6 or earlier contains some bugs in the definitions of the objects. This can result in displaying wrong DNs. The certs are correct! The last major change in the definitions was at March the 28th in 2002.

1.2.2 Database

If you use OpenCA::DB (the default) then you can ignore this. If you use a SQL-database then you should read this perhaps. You can configure your database directly via etc/servers/DBI.conf or you set the options during the installation. You must not set all options for all databases. Some databases locate things like the host or the port automatically by their own configurationfiles (e.g. Oracle or DB2).

1.2.2.1 Troubleshooting

The document contains no data.

This can happen if you forget to install a perlmodule or a databasedriver itself or there is misconfiguration. You need at minimum the following:

- DBI
- DBD::DB_Vendor
- a driver of the databasevendor must also be installed normally

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.

1.2.3 Distinguished Name

The following options influence the DN:

- OrganizationUnit (1.OU, 2.OU ...)
- Organization
- Country
- Locality
- SET_REQUEST_SERIAL_IN_DN and REQUEST_SERIAL_NAME enforce the inclusion of the request's serial in the DN
- SET_CERTIFICATE_SERIAL_IN_DN and CERTIFICATE_SERIAL_NAME enforce the inclusion of the certificate's serial number in the DN (strongly recommended)
- DN_WITHOUT_EMAIL (e.g. S/MIME v3)
- the basic CSR will be described extra because this influence only the requestgeneration and not the certificategeneration directly

Please notice:

OpenCA use internally the DNs like described in RFC 2253. This is the oppoiste order like OpenSSL uses. The transformation from RFC 2253 to the order which OpenSSL uses is handled fullautomatically.

1.2.4 LDAP

The LDAP-interface is only present at the RA. So you must only configure the file of the appropriate RA. The default name of the configuration file is etc/lib/ra.conf.

blablabla

1.2.4.1 Troubleshooting

Bind failed. Errorcode 49.

These messages signal that the credentials for the LDAP-server are wrong. The host and the port are not wrong because the bind-operation is performed after the connect-operation which requires the correct host and the correct port. The things which are wrong can be ldaproot (LDAP-admin's DN) and ldappwd. Please take in mind that the LDAP-server must be initialized.

Connection refused.

This happens if the ldapserver or the ldapport is wrong. Please take in mind that the LDAP-server must be running.

1.2.4.2 Sourcecodeorganization

Structure of the code

```
Scripts for the import |
   of certificates     |
                       |
-----------------------| scripts to add
                       | objects to LDAP
   export-import.lib   |
                       |
---------------------------------------

                ldap-utils.lib
```

The relevant commands

addCertsLDAP (puts all valid certs to LDAP)
addCrlLDAP (puts all CRLs to LDAP)
importAllFromCA (via export-import.lib)
importCRL (via export-import.lib)
importCerts (via export-import.lib)
importCertsLDAP (puts all certs from the last import to LDAP)
importConfig (puts CA-certs to LDAP)
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)

export-import.lib

eximObjectToLDAP

ldap-utils.lib

addCertsUsers (will be moved to importCertsLDAP)
addLDAPobject (takes a cert and create the necessary nodes in the LDAP)
addLDAPattribute (add certs and CRLs to the LDAP)

1.2.5 Configuration of basic CSR

The basic CSR is the most flexible way in OpenCA to create CSRs. Let's start with an example:

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

DN_TYPES "BASIC"

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_2 "Name"
DN_TYPE_BASIC_ELEMENT_3 "Certificate Request Group"
DN_TYPE_BASIC_ELEMENT_3_SELECT "Internet" "Partners" "Employees" "Trustcenter"
```

The first line defines the available keysize. The next variable DN_TYPES defines the available configurations of basic_csr. The command basic_csr is called via a link and the link must contain an option CSR_TYPE which defines the configuration which is used for this CSR.

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 generated. If the value is N or NO 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.

The BASE is the part of the DN 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 DN.

The ELEMENTS is the part of DN which can be defined by the user. The ELEMENT_numbers 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). 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.

1.2.6 RBAC

Please read the whole chapter about RBAC.

1.2.7 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.

1.3 RBAC

The RBAC-code is one of the most difficult parts of OpenCA. Before I start with the technical details I describe the logic behind the details.

1.3.1 Logic

Please read the description of the following five basic parts of the role based access control of OpenCA carefully. They explain the whole concept o which is used for OpenCA. We don't use attribute certificates because they are not supported by the todays software.

1.3.1.1 Roles

The roles are part of every role based system. OpenCA defines several default roles which you can simply extend by other roles which you need. Every certifcate 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.

1.3.1.2 Modules

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 name "RA 1". The details of the configuration are described later. The access rights must be defined for every module again (except they have all the same name).

1.3.1.3 Operations

An operation is a type of action which can be done on the system. An operation can be "csr approve" which means that a certificate service request will be approved. These are onl logical operations.

1.3.1.4 Scripts

The scripts are the files which are placed in lib/cmds/. Every script has a configurationfile 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.

1.3.1.5 Rights

The rights build the ACL for OpenCA. An access right consists of four things:

- the module which is used
- the operator which works on the module (the operator is a role)
- the operation which will be performed
- the owner of the affected module (the owner is a role)

1.3.2 Technical details

1.3.2.1 Algorithm

The different configurationfiles are stored in etc/rbac/

- modules/
- operations/
- rights/
- roles/
- scripts/

The files in modules/ contains nothing. The decoded filename is the name of a role. The files in operations/ and roles/ are used in the same way like the files in modules/.

The files in scripts/ contain four variables:

- SCRIPT

- OPERATION
- OWNER_METHOD
- OWNER_ARGUMENT

There are six OWNER_METHODs:

1. CERTIFICATE_SERIAL
   This method is used if an operation affects a certificate and the role should be detected by the serial of the certificate.
2. REQUEST_SERIAL
   This method is used if an operation affects a CSR and the role should be detected by the serial of the CSR.
3. 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.
4. CGI
   The use of this method is not recommended because the role is not protected by any cryptographic mechanisms.
5. ANY
   The operator must have the right to perform this operation for every role.
6. <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*.

The configurationfile end with the suffix ".conf". A second file with the suffix ".sig" contains a signature of the configurationfile to protect the file against manipulation during transport.

The files in rights/contains the signature of the filename to protect the right. The filename itself consists of the for encoded parameters module, operator, operation and owner. The diffent parts of the filename are seperated by a single "-".

1.3.2.2 Digital Signatures

Actually there is a discussion about the signing of the files in scripts/ and rights/ because the signing costs a lot of performance and we don't really improve the security with the signing.

1.3.2.3 Modified Base64

We use a modified Base64-encoding for the filenames. We replace "/" by "_". This was necessary because Unix interprets "/" as a directorychange and the escaping of such special characters is different on several platforms.

Chapter 2
Operatorguide

2.1 Introduction

Here we have to include lifecycle.ps if somebody knows how to do this with TeXmacs.

2.2 Initialization

The initialization consists of three phases. The first phase initializes the CA itself, the second phase creates the first user-certificate and the last phase creates the first certificate for a web-server.

2.2.1 Initialize the certification authority

The following steps are necessary to setup the PKI, so not all steps are cryptooperations.

1. DB Setup
   Here you have only to click on the link and then the configured database will be initialized. This action will remove all data from the database if the database already exists. (Only SQL-databases are protected against the destruction of an existing database.)
2. Keypair Setup
   This is the first cryptooperation. You can choose the length of the private key and you must enter the symmetric encryptionalgorithm and the passphrase to encrypt the new private key. The private key is an RSA-key. We don't support DSA today.
3. Request Setup
   The CSR for the CA-certificate can be generated by this link or you can create the request by hand. If you use the link then OpenCA will ask you for the components of the CA's distinguished name and the passphrase of the CA's private key to create the CSR.
   If you want to create the request by hand then you must simply create the request and store them in OPENCADIR/var/crypto/reqs/careq.pem.
4. Certificate Setup
   The setup of the CA-certificate allows to general ways. You can create a self-signed CA-certificate if you want to create a Root-CA or you can export the CSR and import the CA-certificate if your CA is a sub-CA.
5. Final Setup
   The last step is used to create the certificate chain which is necessary for the verification fo digital signatures and to export the whole configuration. This configuration you must import into the RA before you start using the RA in production.

2.2.2 Create the initial administrator

The initial administrator certificate is the first user-certificate. You should use the role CA Operator for this certificate because this is the certifcate of the operator which issues the first certificates.

1. Create a new request
   Simply enter all the data which are requested and decide which keylength you want. Please notice that many hardware security modules (e.g. several smartcards) only support keys with a maximum length of 1024 Bit. If you want to store the private key and the certificate of the initial CA Operator on a smartcard please check the available keysizes before you get into trouble.
2. Edit the request
   Now you have the last chance to modify the distinguished name of the certificate. You can fix the role and set the correct Subject Alternative Name. The DN uses the order which is described in RFC 2253.
3. Issue the certificate
   Simply click on issue Certificate to create the new certificate. You can use the button on the page which is displayed after the editing of the request too.
4. Handle the certificate

This page is also available via Information –> Certificates –> Valid Certificates. Here you can download the certificate and the private key.

2.2.3 Create the initial RA certificate

The initial RA certificate is the certificate for the https-server. You should use the role Web Server for this certificate.

The steps are the same like for the initial administrator.

2.2.4 Troubleshooting

If I click on edit request then I see an errormessage.

If the errormessage is *Request not present in DB or the status of the request was changed!* then you start the intialization without a re-intialization of the database. The initial request has not the serial 1 and the link is wrong. You must start again with the initialization of the database but you must not create the private key and the request for the CA again.

If I click on issue Certificate then I see the errormessage *pwd.html not found*.

This message appears if you configured the wrong path for the apache's Document-Root. This can happen in httpd.conf or by using a wrong value during ./configure. Please contact your administrator or read the administratorguide for more information.

I exported the configuration (or all) from the CA and my certifcates are deleted.

This is normally not the exact discription. You exported the configuration (or simply all) and then you start initializing the RA but the CA and the RA are on the same machine with the same configuration. The first step you are doing is initialize database.

This will initialize the database but the CA and the RA use the same database. If you use DBM-files (DBmodule "DB") then you overwrite in this moment the CAs database too. If you use a SQL-database (DBmodule "DBI") then nothing is happen and you get only an errormessage that the initialization on the RA fails.

Cannot convert PEM-certificate and PKCS#8-key to PKCS#12-formatted file!

This messages occurs if you try to export a private key and the associated certificate to a PKCS#12-file. You must use the passphrase which you entered in the two PIN-fields during you make the request. You must not enter the CA's passphrase here!

2.3 Export and Import of data

The export and import of data is necessary to exchange the data between the CA which is offline and the rest of the trustcenter. Actually we support the fillowing functions:

1. From CA to RA
   - CA-Certificate(s)
   - Configuration

     These functions supports the exchange of the CA-certificates and the RBAC-related data.
   - Certificates
   - CRLs
   - All

     This functionality should normally used. It exports and imports all relevant data. So you must not care about anything by yourself.

2. From RA to CA
   - CSRs
   - CRRs
   - All

     This functionality should normally used. It exports and imports all relevant data. So you must not care about anything by yourself.

The main problem today is that we must export and import all data at every time because there is no verification mechanism for the export and import which checks the successful transport of objects.

2.4 CSR handling

The handling of a certificate service request consists of four steps:

1. Editing
2. Approving
3. Export/Import
4. Issuing the certificate

2.4.1 Editing the CSR

2.4.1.1 Distinguished Name

The distinguished names which are displayed and sometimes editable in OpenCA uses the order defined in RFC 2253. All conversions are handled automatically. A sample DN could be "cn=John Doe, ou=public relations, o=Joe's Pub, c=uk".

2.4.1.2 Subject Alternative Name

The subject alternative name is the place where the emailaddress or the DNS-name can be stored. OpenCA uses the emailaddress by default because the emailaddress should not be part of the DN of a usercertificate (see RFC 2633 "S/MIME Version 3 Certificate Handling" section 3).

The subject alternative name must use the format defined by OpenSSL. The format looks like follows:

```
SubjectAltName ::= GeneralNames
GeneralNames ::= GeneralName ["," + GeneralName]*
GeneralName ::= CHOICE {
        rfc822Name,
        dNSName,
        uniformResourceIdentifier,
        iPAddress,
        registeredID}
rfc822Name ::= "email:" + IA5String
dNSName ::= "DNS:" + IA5String
uniformResourceIdentifier ::= "URI" + IA5String
iPAddress ::= "IP" + OCTET STRING
registeredID ::= "RID" + OBJECT IDENTIFIER
```

2.4.1.3 Role

The role is important for the extensions and for the logical meaning of the certificate.

2.4.2 Approving the CSR

You must simply view at the request and if you think that all relevant items are correct then you can use one of the approve buttons. If you use Approve and Sign then you can sign and approve the request with a Netscape 4.7x or IE 5.0+. If you cannot or must not sign the request then you can use the button Approve without Signing. The data of the request cannot be changed after the request was approved.

2.4.3 Export/Import of the CSR

If you use an offline-CA then you must export the requests from the RA and import the requests into the CA. More details you can find in the section *"Export and Import of data"*.

2.4.4 Issuing the certificate

Before you issue a certificate on the CA please check the displayed data carefully. This is the last chance to delete the request. If you want to issue the certificate simply click on Issue certificate and enter the passphrase of the CA. The new certificate will be displayed.

2.4.5 Renew a CSR

This is actually not implemented.

2.4.6 Troubleshooting

General Error Trapped 6206: Cannot build PKCS#7-object from extracted signature!

This can happen if the databases of your browser which store the certificates and keys are corrupt. This is possible if you are adding a new CA-certificate but use the same name like for an already existing CA-certificate (e.g. you made your second testinstallation but forgot to remove the old CA from your browser).

The second possibility is that you have not imported the CA-chain. If you imported the hole chain then you must trust at minimum the root-CA. Netscape doesn't accept untrusted CA-certificates for signing and Netscape needs the hole CA-chain for signing.

You have problems with IE and signing?

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/capicom_start_page.asp

Cannot encrypt PIN-mail! Aborting!

This problem can have two reasons. The encryption of the PIN-Mail can fail if you use some really unstable OpenSSL-snapshots but the common mistake is an unconfigured SERVICE_MAIL_ACCOUNT. You can configure the mailaddress in ca.conf.

2.5 CRR handling

The general way for a CRR is the same like for a CSR. Therefore we only discuss the troubleshooting here.

The only different between a CSR and CRR is that an Operator can start a revocation on the RA. If OpenCA view a certificate and you see the button revoke then you can start a revocation.

2.5.1 Troubleshooting

General Error Trapped 6206: Cannot build PKCS#7-object from extracted signature!

This can happen if the databases of your browser which store the certificates and keys are corrupt. This is possible if you are adding a new CA-certificate but use the same name like for an already existing CA-certificate (e.g. you made your second testinstallation but forgot to remove the old CA from your browser).

You have problems with IE and signing?

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/capicom_start_page.asp

2.6 CRLs

Here we have to insert some data ...

2.7 Batchprocessors

OpenCA has several very powerful batchprocessors to support big organizations with certificates and don't causes a lot of money. There are two categories of batchprocessors. One class of batchprocessors should automate the work of the CA Operators and the other class of batchprocessors should automate the work of the RA Operators.

Nevertheless all batchprocessors are installed on the CA by default because the use of nearly fullautomatic software is a high risk.

2.7.1 CAO batchprocessors

2.7.1.1 Issue certificates

If you have a big organization with many RA Operators which uses their certificates and private keys to approve and sign the certificate requests of your members then it is perhaps not necessary to have a CA Operator whom checks every request again.

The main idea is that the CA Operator only enters the following data:

- the role of the RA Operator(s)
- the requested role of the certificate request
- the passphrase for CA's private key

The batchprocessor issue every certificate from a request where the requested role matches the specified role, the signing RA Operator has the specified role and the signature is correct.

The usage of this automatic system is not a high risk if you can trust the signatures of your RA Operators.

2.7.1.2 Revoke certificates

The main difference between this and the above described processor is that this processor uses the role of certificate which is assigned to the processed CRR.

2.7.2 RAO batchprocessors

These software components has a complete other focus than the CAO batchprocessors. The idea is that you have some ERP-like systems in the background and by this way you have a defined base of users. Now you need a simple way to enroll a PKI.

Below we describe some batchprocessors but they are only examples. You can find the code in your OpenCA-directory (default: /usr/local/OpenCA/lib/cmds/bp*). If you want to implement another businessprocess simply do it.

2.7.2.1 New users

Let's start enrolling a PKI!

Import new users

First we must import the new users so we need a file on the standard importdevice. The name of this file is specified in /usr/local/OpenCA/etc/servers/ca.conf and the option is BP_File_ImportNewUser. This file must have the following format

```
ID 1234567890
DN CN=testuser, OU=Research, O=OpenCA, C=DE
ROLE User
SUBJECT_ALT_NAME email:testuser@openca.org
STATUS_MSG Does somebody know this guy?
<blank line>
<next dataset>...
```

The ID is a freestyle but unique ID generated by your ERP- or IT-systems. It is required. The DN is an OpenSSL-like DN which is used for the certificate request. OpenSSL-like mean that the attributes must be conform to the definitions in include/openssl/objects.h. Please take in mind that OpenSSL is **casesensitive**. The DN is required. The role specifies the role of certificate which should be issued for the user. It is required.

The SUBJECT_ALT_NAME is OpenSSL-style string which conatins the setting for subjectAltName in openssl.cnf. So if the specified role requires a subject alternative name then you must specify this option.

All other added options are stored in a file. So you can find you message in the file /usr/local/OpenCA/var/batch/1/2/3/4/5/6/7/8/9/0/STATUS_MSG.

A empty line signals a new dataset.

Import permission for new request

Her you import a file which has the same format like the above described. The option in ca.conf is BP_File_ImportNewCSR. The important difference is that only the line with ID is used. All other option will be ignored. So you can use the same file only with different names if your all new users should get immediately a certificate.

Create PINs

Now we need some PINs for our new users to be able to handle their certificate requests automatically.

Export PINs

The generated PINs must be exported to your system to allow you to send some PIN-mails. The exported file has the format:

```
ID 1234567890
PIN kasldfjhfhkljahaf2e32
<blank line>
<next dataset>
```

You can delete the PINs after you send the mail because the CA has a copy of the PIN and the hashed PIN.

Approve requests

The user go with his PIN and his ID to a terminal and requests a certificate via the public gateway. He must enter the PIN in the PIN-field and the ID into the field for the name of the user. The batchprocessor will check the ID, PIN and requested role, add the correct DN and if specified the subject alternative name and sign the hole request.

Delete not hashed PINs

After a request was approved we need no longer the PINs. So simply remove them.

It is possible to mix the steps because they are safe against such *mistakes*. Did you forget 100 users? No problem, simply build a new file and start again (at minimum you must do the steps again which you don't perform for the other users).

2.7.2.2 Renewed certificate requests

Sometimes there are users which need a certificate renewal (e.g. the certificates for the students of your university are only valid for one year). So what should you do if you must do so many renewals?

Import renewed requests

You must build a file with the name of the option BP_File_ImportRenewedCSR in ca.conf. The file has the same format like for **Import Permission for new request**. The requests will be automatically renewed.

Approve renewed users

Now you must simply do the same like for **Approve request**.

2.7.2.3 Update data

Sometimes there are users who want to marry. The problem is that you need the man or woman and cannot forbid them to marry. So you must update there data. No problem. simply build a file like for **Import new users** and this batchprocessor updates the data of such persons. If you now start a renewal for this user then he will get a new certificate.

2.8 Backup and Recovery

We must not discuss the necessity of backups. So we can start with the details and we start with a warning:

OpenCA create no backups of the private key!

If you create a backup with OpenCA then all available cryptoobjects in the database will be exported from the database. The backup will be written to the standard export-device of OpenCA.

If you have a crash and you want to recover then you must first restore the private key and then you must know what do you use for a database. If you use DBM-Files (the default) then you must use the importDB-functionality of OpenCA. This will restore all the objects in your database. If you use a SQL-database then you must use the function replayLog. The SQL-databases include a log where all write-actions are documented so OpenCA can replay all actions which ever happens to the module. The function importDB works for SQL-databases too but replayLog is much better.

Chapter 3
Userguide

3.1 CA-Certificate

3.2 Certificate Service Requests

3.2.1 Basic CSR

3.2.2 Microsoft Internet Explorer

3.2.3 Netscape Navigator and Mozilla

3.2.4 PKCS#10 CSRs

3.3 Certificate Revocation Requests

3.4 Certifcate Revocation Lists