CSI Security

Nokia Siemens Networks
Security

Certificate Authority for Network Elements

Initial Configuration Guide

Version 1.0

Table of Content

1. Introduction

1.1 Executive Summary

Nokia Siemens Networks provides a cost-effective and easy-to-use Public Key Infrastructure (PKI) solution for issuing and managing digital certificates. The management of certificates is automated and compliant to standards.

The described Public Key Infrastructure is based on the INSTA DefSec Oy Certifier® software product. INSTA DefSec is part of the INSTA group (http://www.insta.fi/en) and has signed official and exclusive partnership with NSN, for joint development of a TelCo environment capable PKI.

In this document you find instructions on how to configure the Insta Certifier Application for the following supported blueprint setups, described in the HAPF 2.0 Architecture Description:

- A simple single-Server setup to use for basic testing.
- A simple HA setup (2BE) with minimum traffic separation.
- A security-scaled HA setup using multi-layered traffic separation.

1.2 Scope of this document

NSN PKI solution is described and delivered in a strictly defined HW/SW environment. No deviations to vital OS configuration principles are supported. Installation of HW and Platform is described in the installation guide and not in the scope of this document

However, when it comes to configuring the actual Insta application and PKI functionality, a wide range of functions is possible and its flexibility and ease of administration is the main value proposal of the solution. It is therefore important in trials and demonstrations, to show, that the system can be brought into use fast and easy.

In order to help operators and users to quickly set up a working PKI in a TelCo environment and to e.g. issue certificates for LTE eNB in a SON/PnP solution, this document intends to give a rough guideline on how to set up basic functionality of the Insta application and shows how to perform basic tasks.

This document shall explicitly not be understood as a binding requirement to follow or as “the one and only way” to set up the Insta Certifier. Much more, it is a targeted “recipe” to set up the application and to issue certificates, so that projects can start without delay and optimization of the certifier might take place at any later point in time.

When starting with an “empty” installation, and after following all steps described here, the CA will be issuing certificates and can be used e.g. for LTE transport security or other demos.

1.3 Workflow through the document

Depending on the BTS technology for which certificates should be issued, different steps are necessary to configure Insta PKI or other servers in the environment. Below diagram gives a rough outline of the steps to be taken for each technology, and the chapter in this paper where the steps are described:

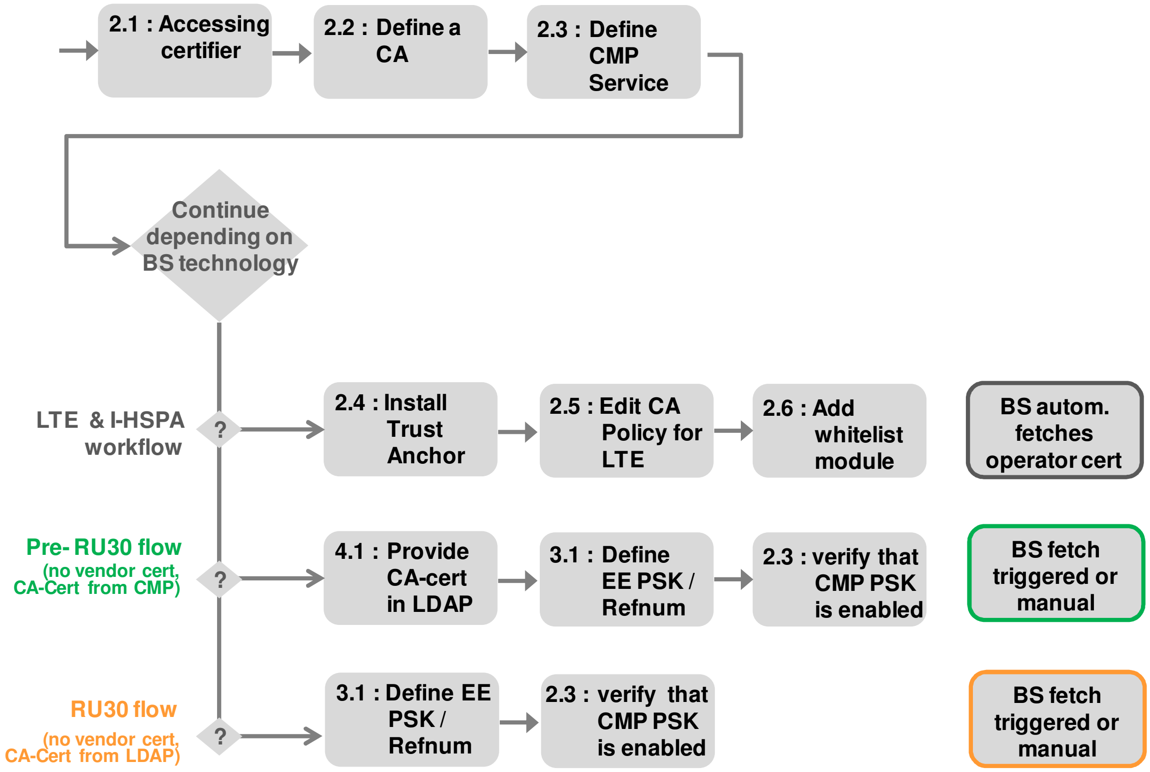

2.1 : Accessing certifier
2.2 : Define a CA
2.3 : Define CMP Service
Continue depending on BS technology
LTE & I-HSPA workflow
?
2.4 : Install Trust Anchor
2.5 : Edit CA Policy for LTE
2.6 : Add whitelist module
BS autom. fetches operator cert
Pre- RU30 flow
(no vendor cert, CA-Cert from CMP)
?
4.1 : Provide CA-cert in LDAP
3.1 : Define EE PSK / Refnum
2.3 : verify that CMP PSK is enabled
BS fetch triggered or manual
RU30 flow
(no vendor cert, CA-Cert from LDAP)
?
3.1 : Define EE PSK / Refnum
2.3 : verify that CMP PSK is enabled
BS fetch triggered or manual

Figure 1: *Accessing Insta Certifier. IP-addr should be of eth0 and default login "admin/admin"*

MIND: "BS fetch triggered or manual" means, that a BS without preinstalled vendor-certificate can't follow an automated (SON/PnP) enrollment process like an LTE BS does. Instead, it must be triggered by an external (i.e. outside the BS and outside certifier application) tool like n3factmx or scripts.

2. Standard PKI Configuration Scenario

Following configuration scenarios are described in this document in “Recipe” style. Descriptions are complementing the Insta documentation, for example, “Certifier Administrators Guide”, “Getting Started with Insta Certifier”.

When following the paragraphs of this chapter all through, a basic configuration should be set up that allows to sign certificate requests from LTE eNBs coming in via CMP.

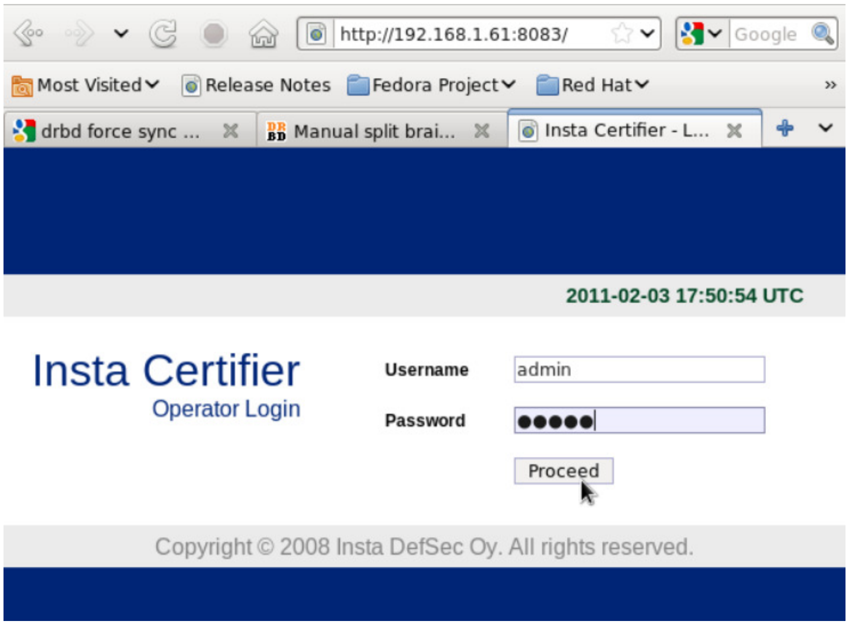
2.1 Accessing the Certifier interface

In order to use Insta Certifier and in order to configure it, you must have access to the command line (via ssh) and to the web-GUI via a browser. Following is a description how to get access to certifier before using it.

2.1.1 Accessing the WebGui

In order to access Insta Certifier in the NSN default configuration after installation, use a browser as depicted below. Connect to the certifier web interface via:

http://[IP_address-or-FQDN]:8083



http://192.168.1.61:8083/

2011-02-03 17:50:54 UTC

Insta Certifier
Operator Login

Username: admin

Password: ●●●●●

Copyright © 2008 Insta DefSec Oy. All rights reserved.

***Figure 2:** Accessing Insta Certifier. IP-addr should be of eth0 and default login “admin/admin”*

2.1.2 GUI Access Prerequisites

Please check each requirement below and correct possible deviations, if you are not able to access the GUI as depicted in Figure 2. You need to have ssh access or console/shell access to the certifier server in order to check the prerequisites:

2.1.2.1 The Insta Certifier Software has been completely installed

Run following commands and make sure to have similar output. If not, your certifier might not be completely installed.

On a Backend machine:

```
[root@cabe-1 ~]# rpm -qa |grep certi
certifier-4.4.0-25
[root@cabe-1 ~]# ll /usr/local/certifier
total 96
drwxr-x--- 3 certfier daemon  4096 Aug  8 11:22 admin-templates
drwxr-x--- 2 certfier daemon  4096 Aug  8 11:24 bin
drwxr-x--- 2 certfier daemon  4096 Aug  8 11:23 conf
drwxr-x--- 3 certfier daemon  4096 Aug  8 11:22 enroll-templates
drwxr-x--- 5 certfier daemon  4096 Aug  8 11:24 lib
drwx------ 2 certfier daemon 16384 Aug  8 10:00 lost+found
-rwxr--r-- 1 certfier daemon 37799 Aug  8 11:22 ssh-ca-setup
-rwxr-x--- 1 certfier daemon  5156 Aug  8 11:24 ssh-ca-start
-rwxr-x--- 1 certfier daemon   934 Aug  8 11:24 ssh-ca-stop
drwxr-x--- 2 certfier daemon  4096 Aug  8 11:23 sybase
drwxr-x--- 8 certfier daemon  4096 Sep 29 13:19 var
[root@cabe-1 ~]#
```

On a Frontend machine:

```
[root@dcnfe-1 ~]# rpm -qa |grep certif
certifsub-4.4.0-25
[root@dcnfe-1 ~]# ll /usr/local/certifsub
total 68
drwxr-xr-x 3 certfier daemon  4096 Aug  8 12:05 admin-templates
drwxr-xr-x 2 certfier daemon  4096 Aug  8 12:05 bin
drwxr-xr-x 2 certfier daemon  4096 Aug  8 12:05 conf
drwxr-xr-x 3 certfier daemon  4096 Aug  8 12:05 enroll-templates
drwxr-xr-x 2 certfier daemon  4096 Aug  8 12:05 lib
drwx------ 2 certfier daemon 16384 Aug  8 10:20 lost+found
-rwxr--r-- 1 certfier daemon 15939 Aug  8 12:05 ssh-ca-setup
-r-xr-x--- 1 certfier daemon  2448 Aug  8 12:05 ssh-ca-start
-r-xr-x--- 1 certfier daemon   222 Aug  8 12:05 ssh-ca-stop
drwxr-xr-x 3 certfier daemon  4096 Aug  8 12:05 sybase
drwxr-xr-x 6 certfier daemon  4096 Sep 29 13:31 var
[root@dcnfe-1 ~]#
```

2.1.2.2 Certifier application is running

You won’t get any connection if the certifier application is not running. Processes on an (optional) frontend are not necessary to get a connection to the web-interface (because the web-interface is part of the certifier-engine / BE) However, in order to avoid later confusion and mistakes, make sure the following is true for your installation (use the same commands, don’t fiddle with the options):

On a Backend machine:

```
[root@pkivma ~]# ps -u certfier -o pid,ppid,euid,cmd
  PID  PPID  EUID CMD
 3373     1   500 dbeng12 -gk all -n certdbeng -s local0
             /usr/local/certifier/sybase/certifier.db -hn 7
 3431     1   500 /usr/local/certifier/bin/ssh-ca-engine -x -p -d 0
             ./conf/engine.conf
 3445     1   500 /usr/local/certifier/bin/ssh-ca-server -x -d 0
             ./conf/server.conf
 3455     1   500 /usr/local/certifier/bin/certifier-snmp-daemon -d
[root@pkivma ~]#
```

On a Frontend machine:

```
[root@dcnfe-1 ~]# ps -u certfier -o pid,ppid,euid,cmd
  PID  PPID  EUID CMD
 3239     1   500 ./bin/ssh-ca-server -x -d 0
             /usr/local/certifsub/conf/server.conf
[root@dcnfe-1 ~]#
```

2.1.2.3 Certifier is actually listening to port TCP:8083

After previous paragraphs make sure that certifier is installed completely and all required processes are running, you should be able to verify, that the certifier admin-server process is actually listening to it's default port 8083.

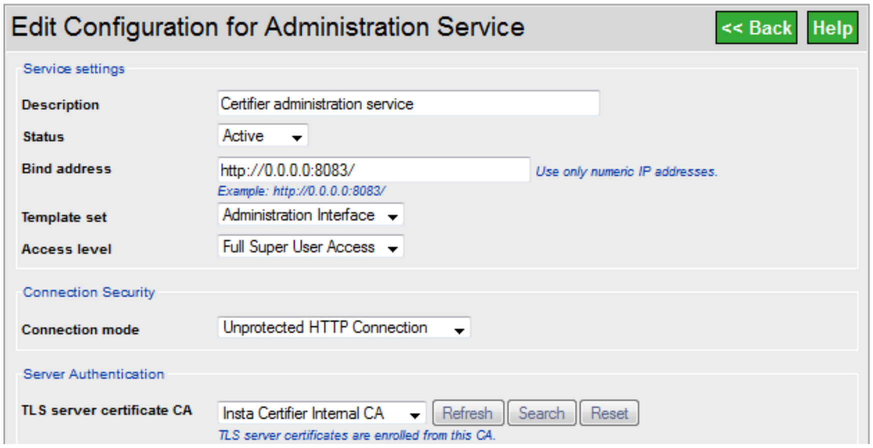
```
[root@cabe-1 ~]# netstat -tunap |grep 8083
tcp 0   0 0.0.0.0:8083   0.0.0.0:*         LISTEN      3501/ssh-ca-server
[root@cabe-1 ~]#
```

2.1.2.4 Certifier is configured to listen to port TCP:8083 on an active IP address

If certifier is not listening to this port, it might be that it has been configured to listen to a different port. This can be, for example, because the admin service has been edited similar to the editing of the parameters for a CMP service in 2.3 (where instead of the CMP URI, it might be that somehow the URI of the Administration Service had been changed to be different from default).

To see if all setting are left to default, you can verify the port 8083 of the Admin Server in the server settings:

- In the main menu, click "Servers", then "Administration Server" to see the list of services that are currently defined on the PKI. Per default, an "Administration Service" and one "Web Enrollment Service" are defined. For service "Administration", click "Edit Service", check the URI under "Bind Address" to be as depicted below:



Edit Configuration for Administration Service

Service settings

Description: Certifier administration service

Status: Active

Bind address: http://0.0.0.0:8083/ Use only numeric IP addresses.
Example: http://0.0.0.0:8083/

Template set: Administration Interface

Access level: Full Super User Access

Connection Security

Connection mode: Unprotected HTTP Connection

Server Authentication

TLS server certificate CA: Insta Certifier Internal CA
TLS server certificates are enrolled from this CA.

Figure 3: *Extract/part of the "Edit Administration Service" menu, "allowing connections to any IP and 8083*

2.1.2.5 Port TCP:8083 is not blocked by iptables

Check that the current iptables policy has port 8083 permanently on ACCEPT:

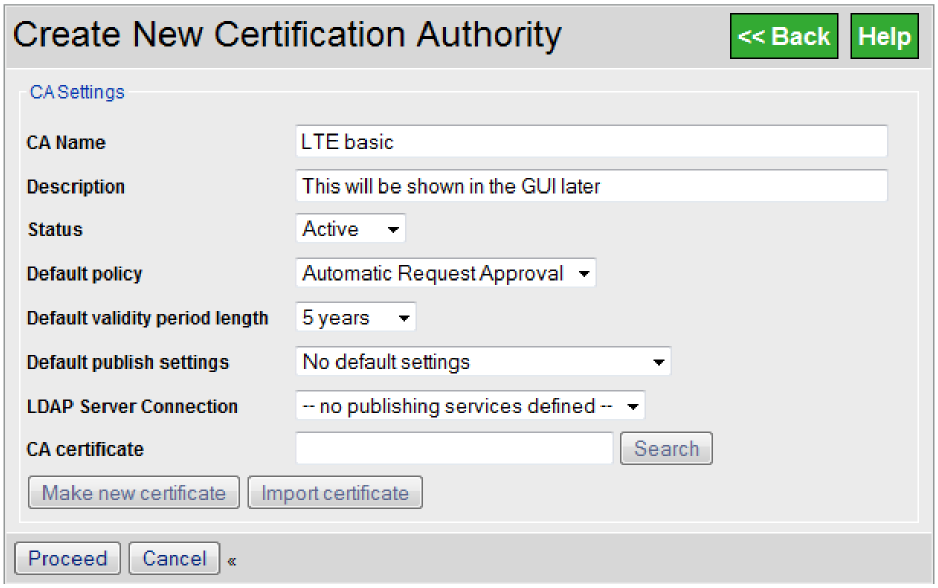
```
[root@cabe-1 ~]# cat /etc/sysconfig/iptables |grep 8083
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8083  -j ACCEPT
[root@cabe-1 ~]#
```

2.2 Defining a CA

Certificates can only be issued by a Certificate Authority (CA). In order to use Insta Certifier, CAs have to be defined. Depending on the operator’s organization and certification practice policy, several CAs or an entire hierarchy of different CAs must be created.

In a simple case, it is enough to create on single CA that can issue certificates for LTE eNBs, as described below. To create a new CA:

- Log in to the Administration Service. (https://<PKI-Admin-Server>:8083)
- On the main menu (left), click CA Hierarchy. CA List is displayed.
- Click the Create New CA button. The Create New Certification Authority page opens (see Figure below (Basic CA entity configuration)).
- Fill in the basic information about the new CA entity.



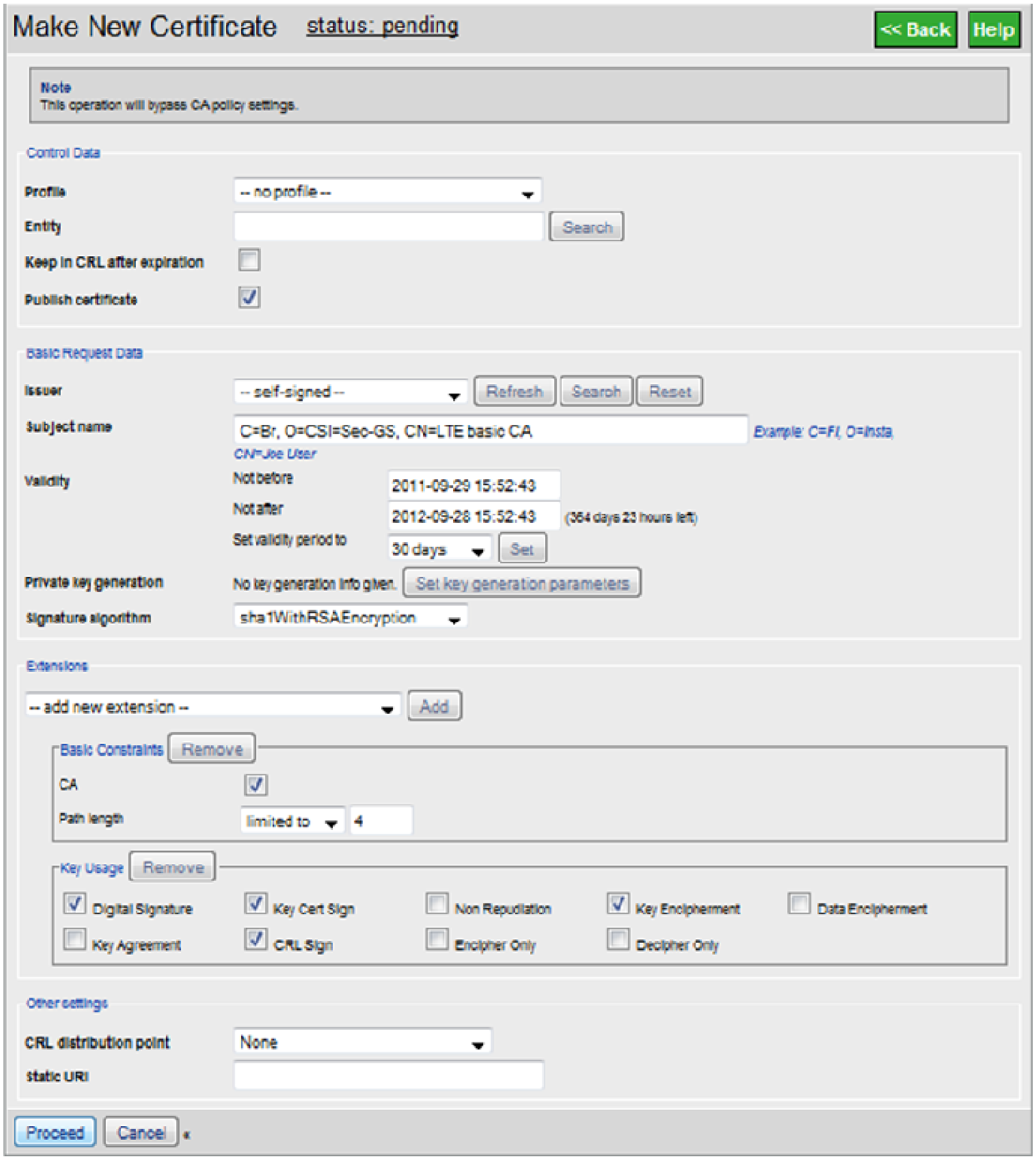
Create New Certification Authority

CA Settings

CA Name	LTE basic
Description	This will be shown in the GUI later
Status	Active
Default policy	Automatic Request Approval
Default validity period length	5 years
Default publish settings	No default settings
LDAP Server Connection	-- no publishing services defined --
CA certificate	

Figure 4: *CA Creation Window, with recommended “get started” settings (Change only Name/Description)*

- Click on Create New CA Certificate option, which will proceed to certificate creation (Figure below (Creating a new CA certificate)). After this, the form will be shown with the CA certificate field automatically set to the newly created certificate. The explanation of each parameter is shown in the following table:



Make New Certificate status: pending

Note
This operation will bypass CA policy settings.

Control Data

Profile: -- no profile --
Entity:
Keep in CRL after expiration: □
Publish certificate: ☑

Basic Request Data

Issuer: -- self-signed --
Subject name: C=Br, O=CSI=Sec-GS, CN=LTE basic CA
Example: C=FI, O=Insta, CN=Joe User
Validity:
Not before: 2011-09-29 15:52:43
Not after: 2012-09-28 15:52:43 (364 days 23 hours left)
Set validity period to: 30 days
Private key generation: No key generation info given.
Signature algorithm: sha1WithRSAEncryption

Extensions

-- add new extension --

Basic Constraints
CA: ☑
Path length: limited to 4

Key Usage
☑ Digital Signature
☑ Key Cert Sign
□ Non Repudiation
☑ Key Encipherment
□ Data Encipherment
□ Key Agreement
☑ CRL Sign
□ Encipher Only
□ Decipher Only

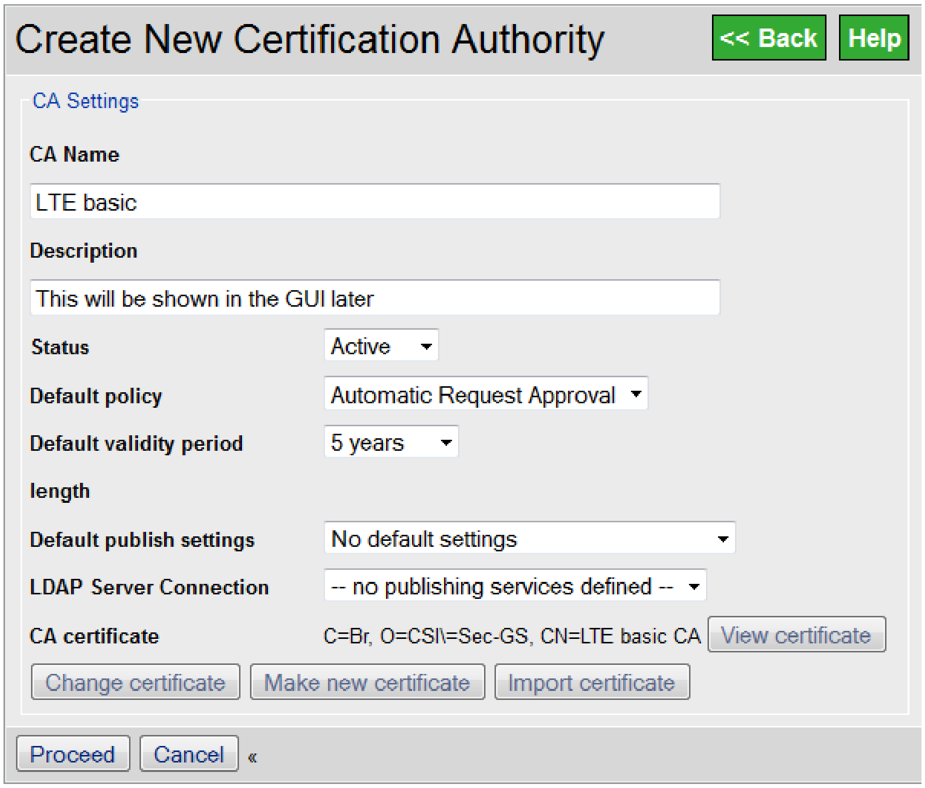
Other settings

CRL distribution point: None
Static URI:

Figure 5: *Certificate creation (during CA creation process)*

Parameter Name	Description
Issuer	Select the Issuer to be self-signed. You can then continue with this CA/Certificate created here. But mind, that according to best practice, self-signed certificates are only used to sign CA certs, and CA's that sign EE (e.g. eNB) certificates should NOT have self-signed certs. When you change the CA cert or it's keypair later on, this will naturally "invalidate" all the EE certs that have been signed with that keypair! (Hence it will require you to establish cross-certification between old/new, to update trust anchors/root CAs in the EE, or to simply re-issue al EE certs)
Subject name	Must be a valid distinguished name (DN) that fits an LDAP data model for use in e.g. publishing. In other words: The LDAP data model used here for the subject name must be such, that you can find the subject later in an LDAP tree if you use, e.g. publishing via LDAP or want to have that CA in a directory service for any other purpose. Following is a simple example: C=<country>, O=<your Organization Name>, CN=< name of CA>.
Validity	Validity period defaults to 1 year from now (current time) on. Common value for a CA cert validity is 5 years.
Private key generation	Click Set Key Generation Parameters to open the Key Generation / Import page. On this page, select the Key Provider Type. A hardware security module (HSM) can be selected if in use (private key storage device in the key provider list) The selected key type and length are now shown in the Public key field on the Make New Certificate page. Longer keys are more secure, but mind that extremely long keys (4096 or more) are not supported by some older applications/systems.
Signature algorithm	The Signature algorithm field default value is SHA-1, it should be used also in the LTE environment.
Basic Constraint	In the basic constraint "Path Length" the number of sub CAs can be determined for the number of entire sub-CA layers that can be configured under this CA (in case sub CA configuration is used). You can set Path Lengths to e.g. 4.
Extensions	The Extensions field contains all extensions in the certificate. The most important is the Basic constraints extension, which must be present in all CA certificates that have the CA flag selected (it is selected by default).

❐ Once all parameters are filled in, press “Proceed” in the “Make new Certificate” Dialogue and “Proceed” again in the “Create new CDertificate Authority” Window as it appears similar like the one below.



Create New Certification Authority

CA Settings

CA Name

LTE basic

Description

This will be shown in the GUI later

Status: Active

Default policy: Automatic Request Approval

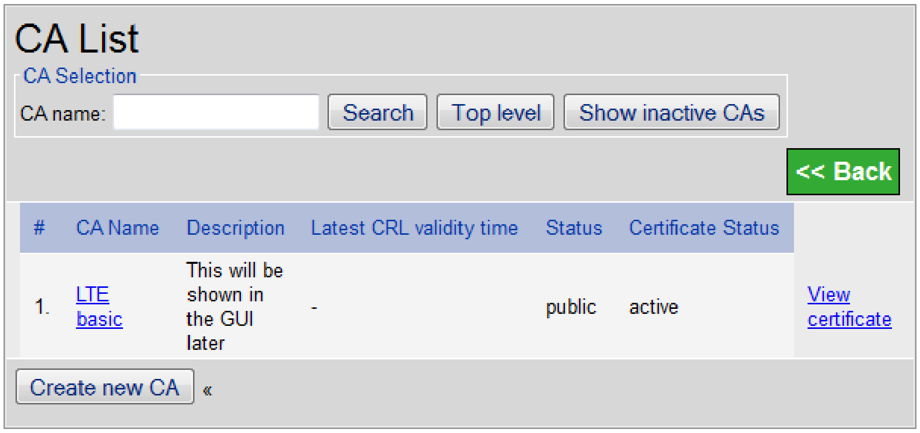
Default validity period length: 5 years

Default publish settings: No default settings

LDAP Server Connection: -- no publishing services defined --

CA certificate: C=Br, O=CSI\=Sec-GS, CN=LTE basic CA

Figure 6: *Final dialogue window before accepting the new certificate with “Proceed”*



CA List

CA Selection

CA name:

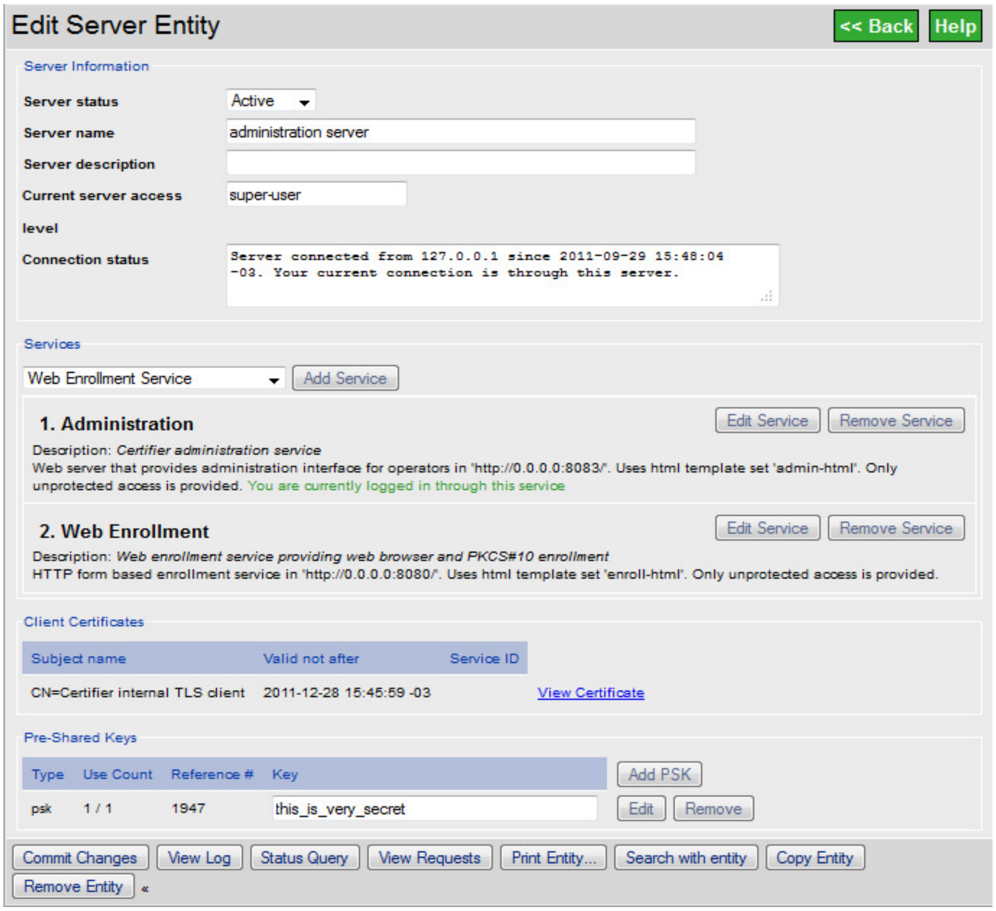
#	CA Name	Description	Latest CRL validity time	Status	Certificate Status	
1.	LTE basic	This will be shown in the GUI later	-	public	active	View certificate

Figure 7: *Created CA will now show in the CA Hierarchy List*

2.3 Defining a CMP service for the CA

The PKI requires services to receive Certification Requests. In LTE environment or other TelCo networks, a CMP service normally needs to be defined. CMP is used to receive and sign certificate requests and send back the actual certificate and the root certificate to an end entity. In order to define CMP service, do the following:

- ❒ In the main menu, click “Servers”, then “Adiministration Server” to see the list of services that are currently defined on the PKI. Per default, an “Administration Service” and one “Web Enrollment Service” are defined. Do never disable the “Administration Service” as this would disable the GUI.



Server status	Active
Server name	administration server
Server description	
Current server access level	super-user
Connection status	Server connected from 127.0.0.1 since 2011-09-29 15:48:04 -03. Your current connection is through this server.

Web Enrollment Service

1. Administration

Description: *Certifier administration service*
Web server that provides administration interface for operators in 'http://0.0.0.0:8083/'. Uses html template set 'admin-html'. Only unprotected access is provided. You are currently logged in through this service

2. Web Enrollment

Description: *Web enrollment service providing web browser and PKCS#10 enrollment*
HTTP form based enrollment service in 'http://0.0.0.0:8080/'. Uses html template set 'enroll-html'. Only unprotected access is provided.

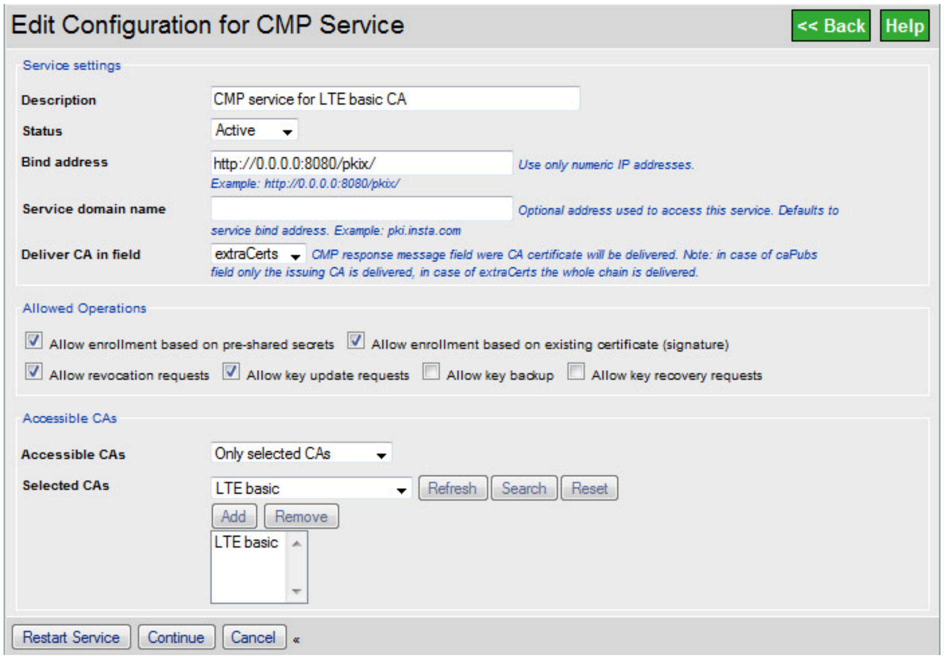
Subject name	Valid not after	Service ID
CN=Certifier internal TLS client	2011-12-28 15:45:59 -03	

Type	Use Count	Reference #	Key
psk	1 / 1	1947	this_is_very_secret

Figure 8: *“Server Entity” screen, from where a CMP service can be added*

- ❒ In the “Edit Server Entity” screen, under “Services”, select “CMP Service” in the dropdown box beside and press “Add Service”. You will get to the dialogue window depicted below.

  Press “Continue” when done with the CMP parameters as shown below.



Edit Configuration for CMP Service

Service settings

Description: CMP service for LTE basic CA

Status: Active

Bind address: http://0.0.0.0:8080/pkix/ *Use only numeric IP addresses.*
Example: http://0.0.0.0:8080/pkix/

Service domain name: *Optional address used to access this service. Defaults to service bind address. Example: pki.insta.com*

Deliver CA in field: extraCerts *CMP response message field were CA certificate will be delivered. Note: in case of caPubs field only the issuing CA is delivered, in case of extraCerts the whole chain is delivered.*

Allowed Operations

☑ Allow enrollment based on pre-shared secrets ☑ Allow enrollment based on existing certificate (signature)
☑ Allow revocation requests ☑ Allow key update requests ☐ Allow key backup ☐ Allow key recovery requests

Accessible CAs

Accessible CAs: Only selected CAs

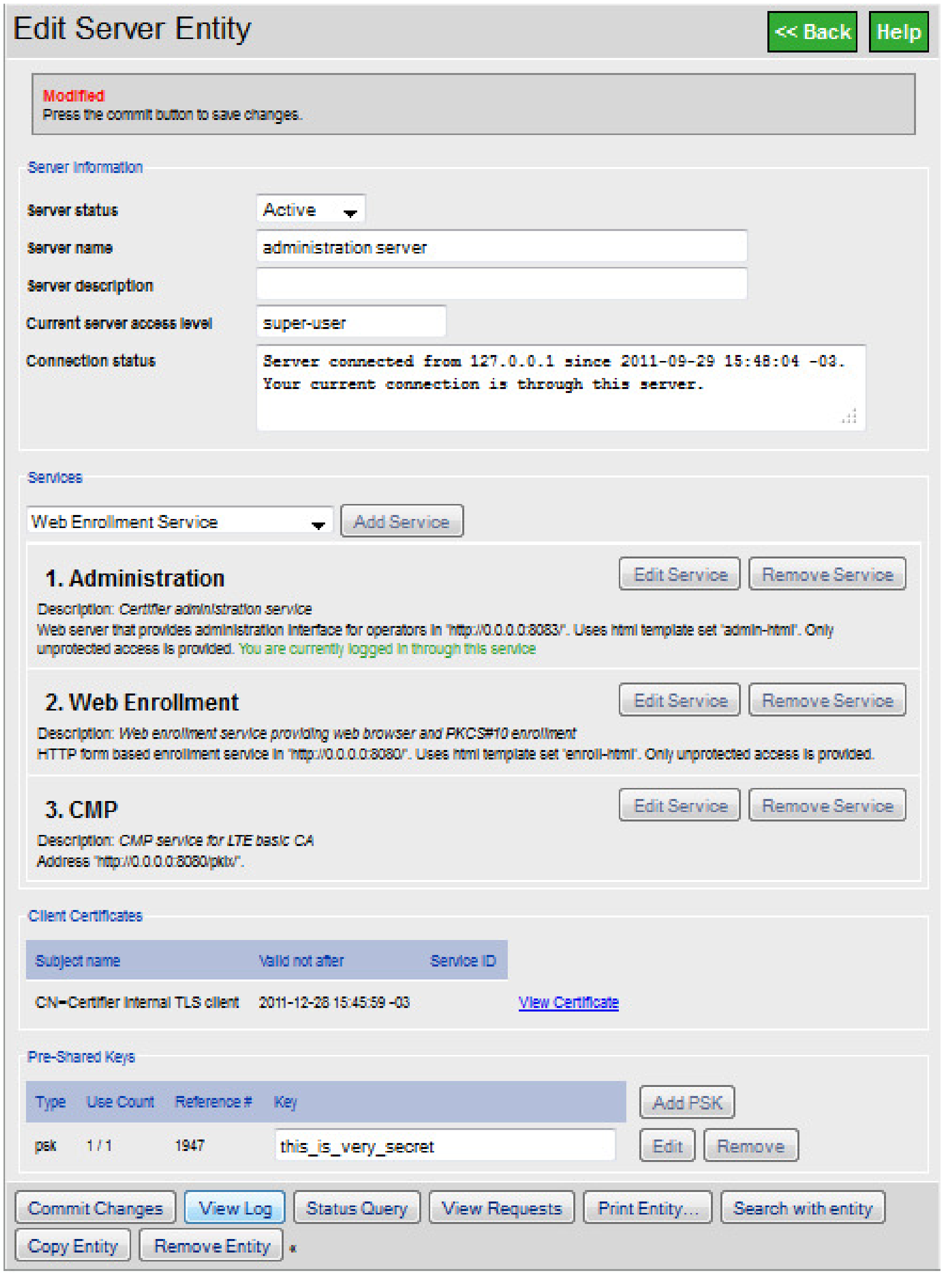
Selected CAs: LTE basic

LTE basic

Figure 9: *Parameters of the CMP service to be added*

Parameter Name	Description
Bind Address	The CMP process will bind to the IP and port as given in this URI. External clients will contact the CMP service by using the complete URI. If IP address is 0.0.0.0 as in the example, CMP service is listening to all locally available addresses on the server.
Deliver CA in Field	The client/EE that requests for signing it’s key does not only need an own certificate in response, but also the cert of the signing CA and eventually all higher layer certs over that signing CA. The field inside the CMP response defines, what and how is transmitted. Use “extraCerts” for RL30/RU30 or later.
Allowed Operations	Define what type of requests this CMP service is allowed to handle. For regular LTE functionality, at least “Allow enrollment based on existing certificate” must be enabled. For 2G/3G and legacy (or non TelCo) applications, it is recommendable to have also “Allow enrollment based on Pre-Shared keys”.
Accessible CAs	Defines, for which CAs the CMP service can accept and handle requests. In the flow of this example, at least the “LTE basic” CA defined in 2.2 should be accessable..

❒ When done with the CMP parameter definition according to the previous picture, press “Continue” and you will get back to the “Edit Server Entity” Dialogue. It has now a CMP service as well and a red notification in the upper left that says “Modified”. Do now press the “Commit Changes” button to accept the CMP service (Then, the “Modified” warning will disappear). Then press “Status Query” and all services including CMP must be “running”



Modified
Press the commit button to save changes.

Server Information

Server status	Active
Server name	administration server
Server description	
Current server access level	super-user
Connection status	Server connected from 127.0.0.1 since 2011-09-29 15:48:04 -03. Your current connection is through this server.

Services

Web Enrollment Service

1. Administration

Description: *Certifier administration service*
Web server that provides administration interface for operators in "http://0.0.0.0:8083/". Uses html template set 'admin-html'. Only unprotected access is provided. You are currently logged in through this service

2. Web Enrollment

Description: *Web enrollment service providing web browser and PKCS#10 enrollment*
HTTP form based enrollment service in "http://0.0.0.0:8080/". Uses html template set 'enroll-html'. Only unprotected access is provided.

3. CMP

Description: *CMP service for LTE basic CA*
Address "http://0.0.0.0:8080/pkix/".

Client Certificates

Subject name	Valid not after	Service ID	
CN=Certifier internal TLS client	2011-12-28 15:45:59 -03		View Certificate

Pre-Shared Keys

Type	Use Count	Reference #	Key
psk	1 / 1	1947	this_is_very_secret

Figure 10: *After Definition of the CMP service and before activating the new service*

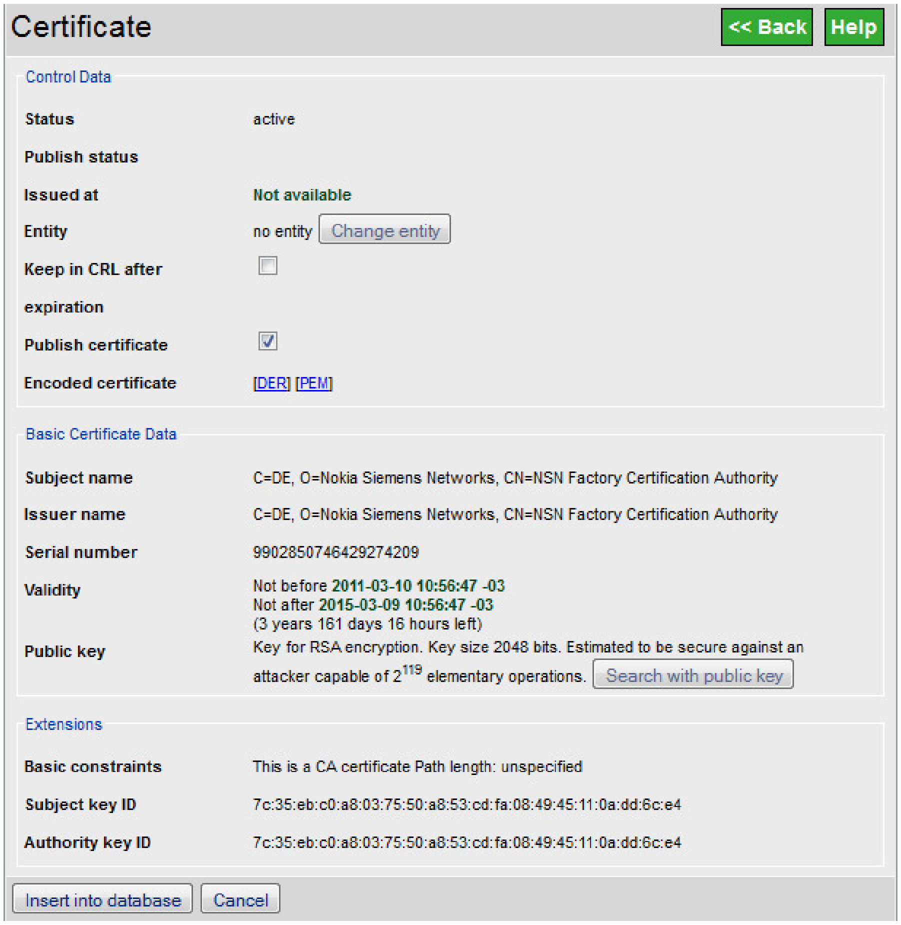
2.4 Installing NSN factory cert as Trust Anchor

A “Trust Anchor” is – simplified – a replacement for cross certifications. If an external CA has signed certificates that should be trusted by Insta certifier (e.g. during an authentication with external certificate), then Insta can verify the external certificates against a “Trust Anchor”. Hence, a “Trust Anchor” is a certificate, which public key is used to verify signatures of certificates that are signed by other CAs outside the own PKI. In LTE, the NSN Factory Certificate has to be installed as a Trust Anchor, because it is used to verify the eNB vendor certificate.

- ❐ In the main menu, press “System Configuration”, then chose “Trust Anchors” (second from top), and “Browse” to find new Trust Anchor files (i.e. certificate file of the NSN Factory certificate) and press “Upload new Trust Anchor”

  MIND: The NSN Factory certificate is to be obtained from the responsible PM/SSM for the eNB delivery!

- ❐ Before you finally add the Trust Anchor, you will see a screen as below. “Insert into Database” will close the operation.



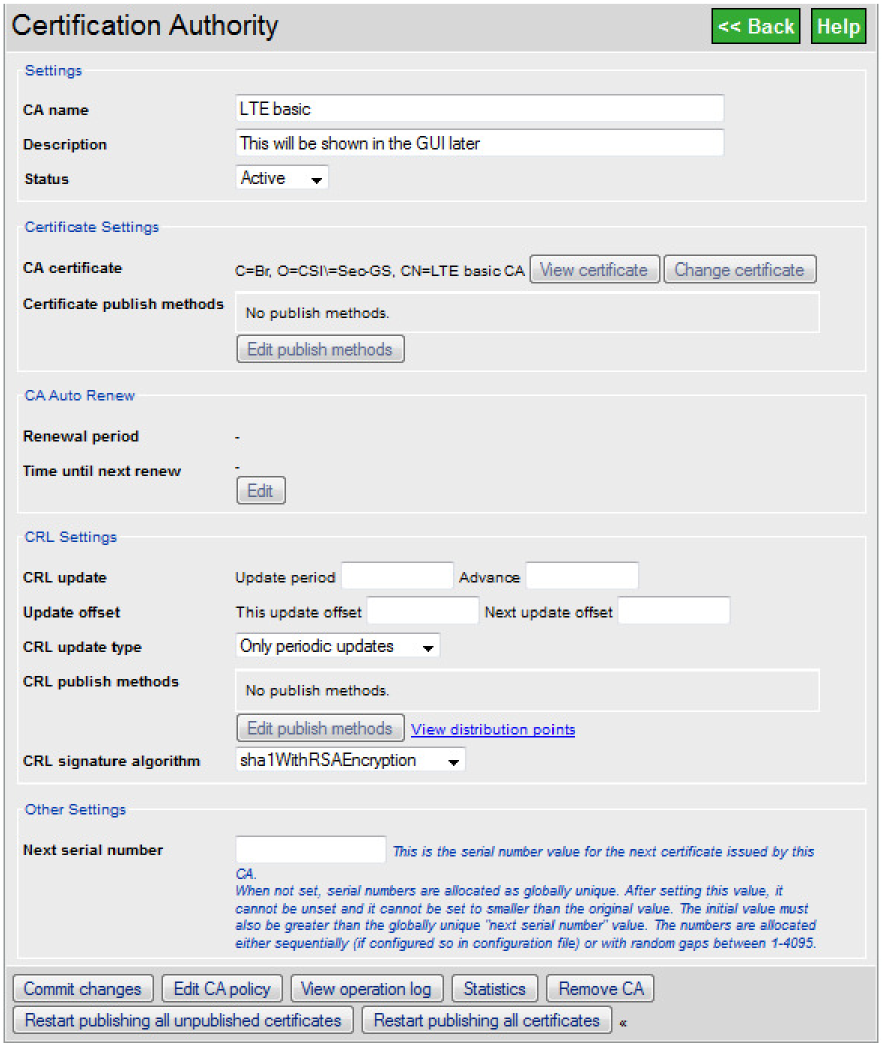
Status	active
Publish status	
Issued at	Not available
Entity	no entity
Keep in CRL after expiration	☐
Publish certificate	☑
Encoded certificate	[DER] [PEM]
Subject name	C=DE, O=Nokia Siemens Networks, CN=NSN Factory Certification Authority
Issuer name	C=DE, O=Nokia Siemens Networks, CN=NSN Factory Certification Authority
Serial number	9902850746429274209
Validity	Not before 2011-03-10 10:56:47 -03 Not after 2015-03-09 10:56:47 -03 (3 years 161 days 16 hours left)
Public key	Key for RSA encryption. Key size 2048 bits. Estimated to be secure against an attacker capable of 2^{119} elementary operations.
Basic constraints	This is a CA certificate Path length: unspecified
Subject key ID	7c:35:eb:c0:a8:03:75:50:a8:53:cd:fa:08:49:45:11:0a:dd:6c:e4
Authority key ID	7c:35:eb:c0:a8:03:75:50:a8:53:cd:fa:08:49:45:11:0a:dd:6c:e4

Figure 11: *Display of the Certificate to be used as a new Trust Anchor*

2.5 Defining a basic CMP policy for LTE

Insta has a “Policy engine” that is made up by chains to which different modules can be added. Each module implements a certain logic, by which a certificate request is handled. The context of the module is given by the chain, e.g. there is a “receive request” chain to define module to define what should be done when a new request is received. The “accept request” chain defines, what to do with an accepted request. Apply the following configuration to ensure, that requests from LTE eNB are handled properly:

- ❒ In the main menu, click on “CA Hierarchy” to list available CAs. Then click on “LTE basic” to see the settings of the CA defined in 2.2, see picture below.



Certification Authority

Settings

CA name	LTE basic
Description	This will be shown in the GUI later
Status	Active

Certificate Settings

CA certificate	C=Br, O=CSI\=Seo-GS, CN=LTE basic CA
Certificate publish methods	No publish methods.

CA Auto Renew

Renewal period	-
Time until next renew	-

CRL Settings

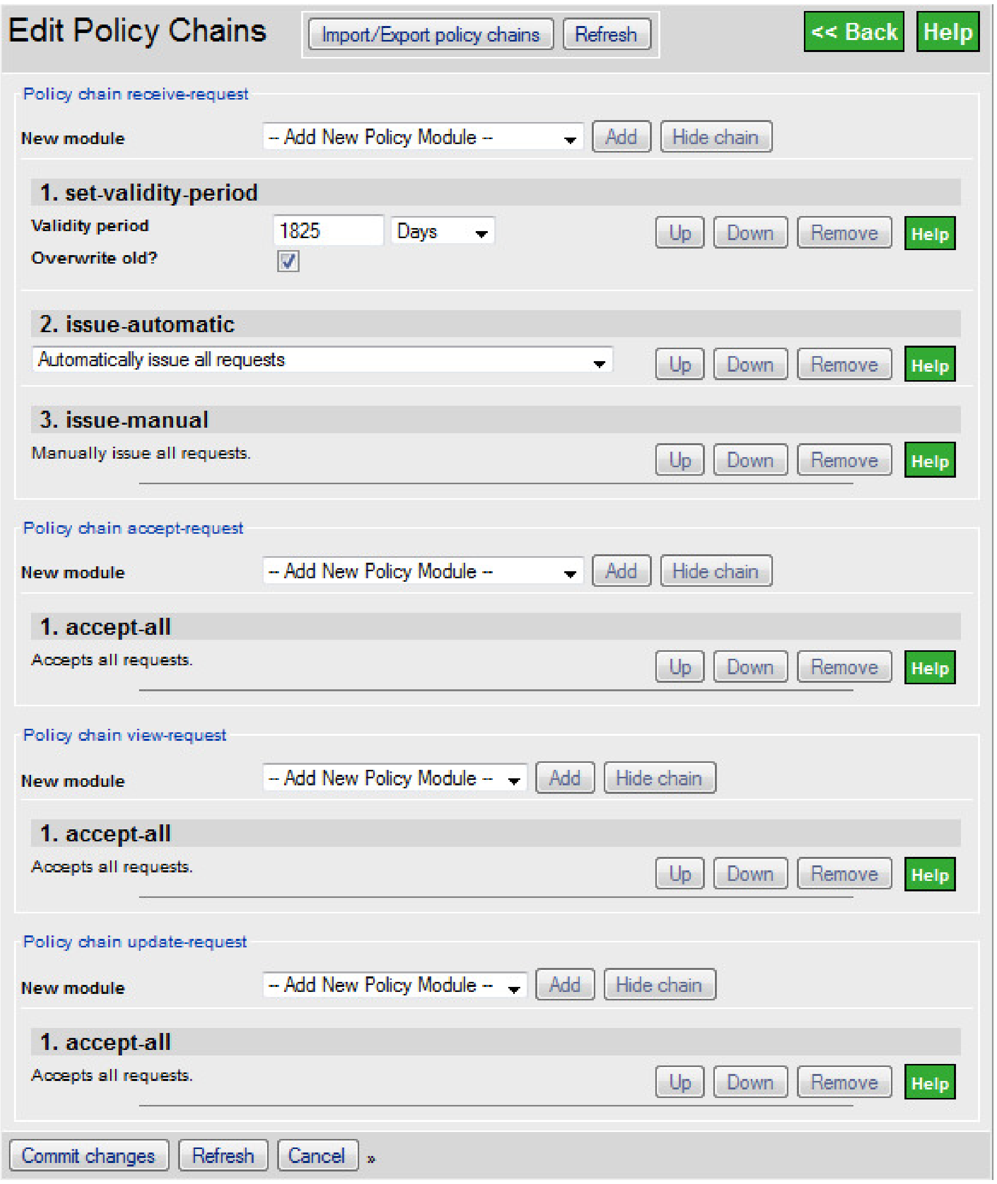
CRL update	Update period Advance
Update offset	This update offset Next update offset
CRL update type	Only periodic updates
CRL publish methods	No publish methods.
CRL signature algorithm	sha1WithRSAEncryption

Other Settings

Next serial number	*This is the serial number value for the next certificate issued by this CA.* *When not set, serial numbers are allocated as globally unique. After setting this value, it cannot be unset and it cannot be set to smaller than the original value. The initial value must also be greater than the globally unique "next serial number" value. The numbers are allocated either sequentially (if configured so in configuration file) or with random gaps between 1-4095.*

Figure 12: *Display of the Certificate to be used as a new Trust Anchor*

- In the Certification Authority screen, click “Edit CA Policy” to get to the „Edit Policy Chains“ screen. Here, if all had been done according to previous descriptions you will see the following:



Edit Policy Chains

Policy chain receive-request

New module: -- Add New Policy Module --

1. set-validity-period

Validity period: 1825 Days

Overwrite old? ☑

2. issue-automatic

Automatically issue all requests

3. issue-manual

Manually issue all requests.

Policy chain accept-request

New module: -- Add New Policy Module --

1. accept-all

Accepts all requests.

Policy chain view-request

New module: -- Add New Policy Module --

1. accept-all

Accepts all requests.

Policy chain update-request

New module: -- Add New Policy Module --

1. accept-all

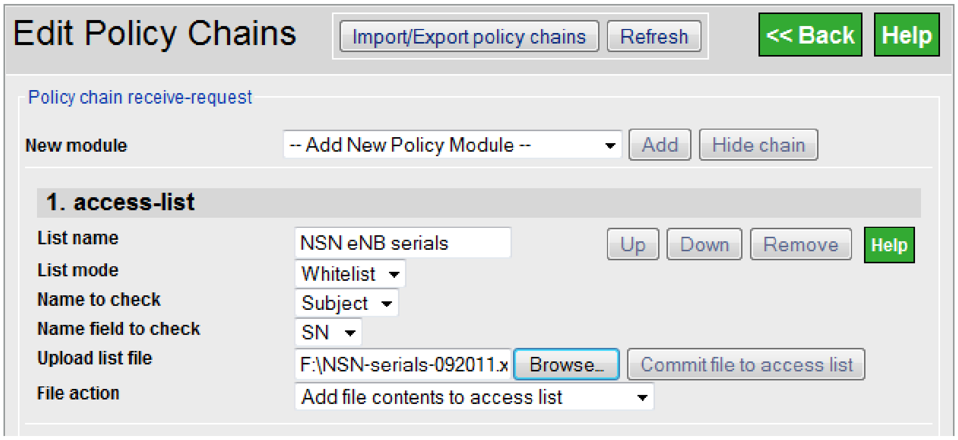
Accepts all requests.

Figure 13: *Policy Chains editing menu*

2.6 Add “whitelist” to CMP policy

With the configuration from the previous *Figure 13*, all incoming requests will be automatically approved. More policy modules at different chains can be added to restrict or otherwise customize this behavior. In LTE, one common change is, to add a whitelist with eNB serial numbers. If that is done, only eNBs who’s serialnumber is part of the certificate subject and also included in the whitelist will receive a certificate. To add a module for whitelist processing, do the following:

- ❒ In the Certification Authority screen, click “Edit CA Policy” to get to the „Edit Policy Chains“ screen. Based on the screen shown in ***Figure 13:*** Policy Chains editing menuFigure 13, do the following addition:
- ❒ In the “New Module” of Policy Chain “receive Request”, select “Access List” from the pulldown box, press “Add” and you will be able to enter parameters as shown below:



New module: -- Add New Policy Module --

List name: NSN eNB serials

List mode: Whitelist

Name to check: Subject

Name field to check: SN

Upload list file: F:\NSN-serials-092011.x

File action: Add file contents to access list

Figure 14: *Policy Chains editing menu, section when adding an “Access List” as whitelist*

NOTE: The Serialnumber (SN) must be part of the certificate subject in the same way as it is listed in the serial-number file. Format of SN-files is a simple XML-tagged list as shown below:

```
<SerialNumberList>
            <SerialNumber>L6094954687</SerialNumber>
            <SerialNumber>L6093734797</SerialNumber>
            <SerialNumber>L1101401122</SerialNumber>
            <SerialNumber>L1101401104</SerialNumber>
            <SerialNumber>L1101401121</SerialNumber>
            <SerialNumber>L1101401103</SerialNumber>
</SerialNumberList>
```

- ❒ After adding the whitelist file, press “Commit Changes” and you get back into the CA configuration Menu. The whitelist will be used now when issuing certs

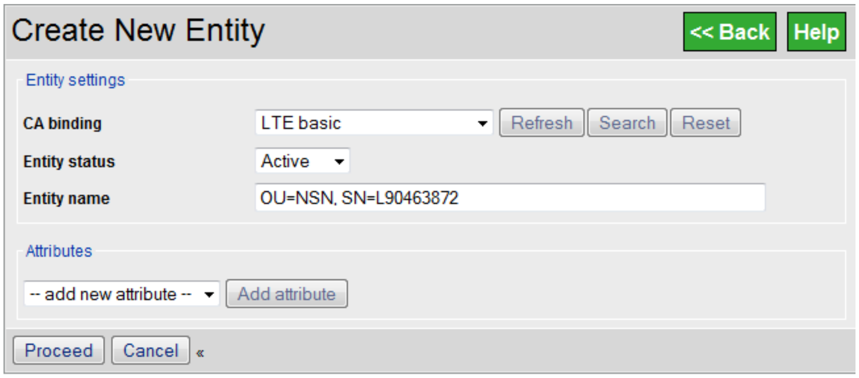
3. Additional tasks to set up Certifier

3.1 Setting up PSK based CMP for 2G/3G

For end entities that are not HSPA or LTE Base Stations with built-in Vendor certificates, an alternative authentication method has to be used to make sure that only valid nodes receive a certificate. The CMP RFC4210 allows two methods: Either to use an external certificate (as done before when using our vendor cert and trust anchor from the factory), or a combination of PSK and reference-number can be used.

In order to use PSK, an end-entity (EE) has to be created first, so that a PSK is created on CA side. Do the following in the Insta GUI to create an EE with PSK:

❒ In the Main Menu on the left side, press “Add New Entity”. You can (no must) bind the EE to a CA from where it must then request it’s certificate, and you can add optional attributes which might (no must) be used when processing the request in the policy chains. After below screen is filled, press “Proceed”



Create New Entity

Entity settings

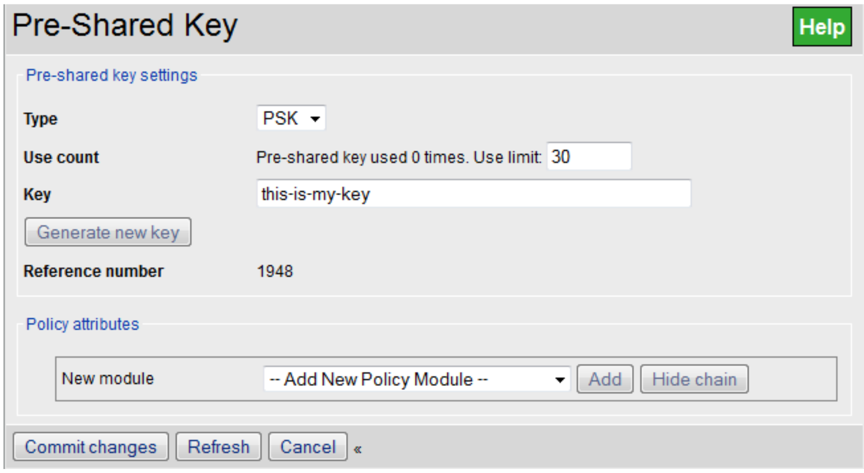
CA binding	LTE basic
Entity status	Active
Entity name	OU=NSN, SN=L90463872

Attributes

-- add new attribute --

Figure 15: *Creation of an Entity (still without PSK)*

❒ Press “Edit” beside the PSK and get to the following screen:



Pre-Shared Key

Pre-shared key settings

Type	PSK
Use count	Pre-shared key used 0 times. Use limit: 30
Key	this-is-my-key
Reference number	1948

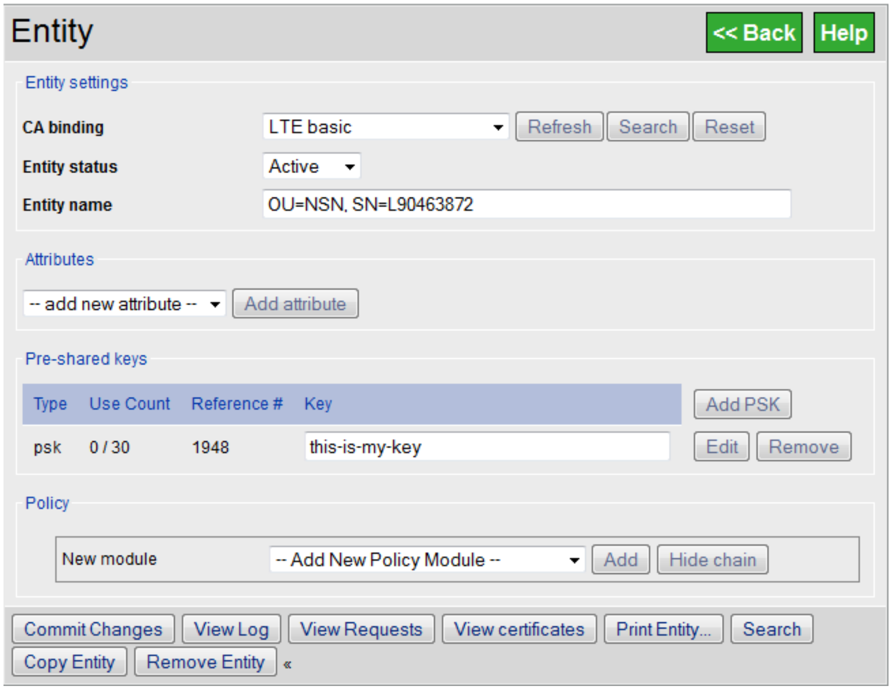
Policy attributes

New module	-- Add New Policy Module --

Figure 16: *Editing PSK parameters of an EE*

- It makes sense to edit the PSK parameters for reasons of e.g.
  - Getting an “easier to remind” (or write down) key! The key will be needed later to enroll for the certificate.
  - Increase the Use-Count. If a key is to be used more than one time, this can be set here accordingly. It means, that in above figure, 30 entities can enroll 30 certificates with different keypairs, but use the same PSK/refnum. (If this is seen secure has to be decided based on the procedures and operator needs)

    After pressing “Commit Changes”, the EE is created with corresponding PSK/refnum and parameters and can be found now in the INsta GUI: To display the EE at any time later, press “Find Entities” in the main menu, aply search filters and proceed. Above entity would be displayed as follows:



Entity settings

CA binding: LTE basic

Entity status: Active

Entity name: OU=NSN, SN=L90463872

Attributes

-- add new attribute --

Pre-shared keys

Type	Use Count	Reference #	Key
psk	0 / 30	1948	this-is-my-key

Policy

New module: -- Add New Policy Module --

Figure 17: *End-Entity (EE) search result if EE is created as described beforehand.*

3.2 Doing basic testing with cmpclient

The ssh-cmpclient program is delivered as part of the Insta SW built and it can play valuable role in testing and verifying e.g. CA policies or basic certifier functionality. It supports both usecases of EE authentication that are already described in Sections 2.5 (authentication of EE with external certificate, in NSN case: vendor certificate) and 3.1 (authentication with PSK/Refnum, in NSN case: legacy BTS). Mind, that in Figure 9 both options must be enabled for the CMP service in order to succeed! EE authentication is a CMP function, not a native CA policy issue!

The ssh-cmpclient program is described in the Insta documentation: “Certifier Reference Guide”, pp. 129 !

3.2.1 Using ssh-cmpclient with PSK/refnum

❒ The program is located in /usr/local/certifier/bin and statically linked, you can “copy” it to any other redhat/linux machine and run it from there. To make execution easier, link it into your path as shown below:

```
[root@pki_sa ~]# ll /usr/local/certifier/bin/ssh-cmpclient
-rwxr-x--- 1 certfier daemon 1175340 May 27 08:40 /usr/local/certifier/bin/ssh-cmpclient
[root@pki_sa ~]# which ssh-cmpclient
/usr/bin/which: no cmpclient in
              (/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/
              bin:/usr/sbin:/usr/bin:/root/bin)
[root@pki_sa ~]# ln -s  /usr/local/certifier/bin/ssh-cmpclient /usr/bin/ssh-cmpclient
[root@pki_sa ~]# which ssh-cmpclient
/usr/bin/ssh-cmpclient
[root@pki_sa ~]#
```

❒ In order to send a certification request with PSK/refnum, you must first create the EE as described in 3.1, or use an existing EE! Before issuing a command to request a certificate with ssh-cmpclient, make sure the PSK/Refnum you use has a spare (i.e. at least one more time) use count!

❒ OPTION / MIND: If your CA policy is set to “Automatically Issue all requests”, then the use-count of the PSK/Refnum does not matter and you can use a PSK/Refnum many times, even if you pass the usecount! This might be useful for testing, it is surely not useful in production!

❒ Run the program as follows to get a certificate based on PSK/Refnum, used parameters are described in the table below, output shown in the screenlog

```
[root@pki_sa ~]# ssh-cmpclient INITIALIZE -P generate://pkcs8@rsa:1024/my-private-key -o
             my-certificate -p 1949:passphrase -s 'CN=my-desired-subject-name'
             http://192.168.1.99:8080/pkix/ 'C=Br, O=CSI\=Sec-GS, CN=LTE basic CA'
Certificate =
  SubjectName = <CN=my-desired-subject-name>
  IssuerName = <C=Br, O=CSI\=Sec-GS, CN=LTE basic CA>
  SerialNumber= 9548
  SignatureAlgorithm = rsa-pkcs1-sha1
  Validity =
    NotBefore = 2011 Oct 12th, 19:30:53 GMT
    NotAfter  = 2011 Nov 11th, 19:30:53 GMT
  PublicKeyInfo =
    PublicKey =
      Algorithm = RSA
      Modulus n  (1024 bits) :
        16285360659969941184961588170834905920804372252319946515939829704681306
        78102801575962148645130507564239024369921775028157023911156503573685513
        29532777463442542024951388721332635058432908851237230339928525589930520
        12213935513316644623864986345010694002348513046725139571406458992007545
        7868311482604344676549997
      Exponent e (  17 bits) :
        65537
    Strength estimation as of July, 2000 considering NFS:
      Attack requires O(2^88) steps, which is roughly  equivalent to
      1.7 * 10^10 years of effort with 1GHz machine.
  Extensions =
    Available = authority key identifier, subject key identifier
    AuthorityKeyID =
      KeyID =
        47:9e:19:f2:30:8f:6d:56:6b:b5:ac:10:17:12:d6:fc:a5:d5:0d:2b
    SubjectKeyID =
      KeyId =
        ae:eb:71:9d:da:d8:b0:66:ed:a3:83:d3:40:11:c0:14:9b:39:6c:c0
  Fingerprints =
    MD5 = 16:46:cf:97:1d:2c:97:87:72:7d:1b:58:6a:5c:59:d5
    SHA-1 = ff:71:6a:12:07:aa:a7:e3:8e:ed:c3:94:3a:72:a2:3a:e5:3a:98:3f
Do you accept the certificate above? y
Accepted user certificate; saving into file my-certificate-0.crt.
Fingerprints =
  MD5 = 16:46:cf:97:1d:2c:97:87:72:7d:1b:58:6a:5c:59:d5
  SHA-1 = ff:71:6a:12:07:aa:a7:e3:8e:ed:c3:94:3a:72:a2:3a:e5:3a:98:3f
Received additional certificate; saving into file my-certificate-extra-0.crt.Fingerprints
              =
  MD5 = 7e:f1:e1:88:30:21:aa:7e:26:aa:af:07:f1:e0:15:a2
  SHA-1 = 5a:4c:d5:21:7f:f4:0a:3b:83:7c:cc:1f:6e:ed:42:99:15:53:55:c5


[root@pki_sa ~]#
```

Parameter Name	Description
INITIALIZE	Triggers a CMP message of type IR
-P generate:…	Generates the private key file (locally) with given parameters (in example: 1024bit RSA key stored as PKCS#8 file ./my-private-key.prv in the current directory. The extension “prv” is added automatically by the system.
-o my-certificate-file	Will store the received certificate as “./my-certificate-file-0.crt” in the current working directory. The extension “-0.crt” is added automatically by the system.
-p 1949:passphrase	Is the Refnum (1949) and PSK (passphrase) connected to the EE for which the certificate is issued

-s 'CN=my-desired-subj…'	Is the desired value of the subjectName attribute in the X.509v3 certificate to be issued. It depends on the CA policy, if the desired name can be taken, must follow certain requirements, or will just be replaced by another subjectName that the CA defines.
http://192.168.1.99:8080/pkix/	Is the URL to which the CMP service is configured in the PKI. The service must be connected to the CA!
'C=Br, O=CSI\=Sec-GS, CN=LTE basic CA'	Is the subjectName of the CA in the CA certificate. By this DN, the CA to issue the certificate is found. The ssh-cmpclient script will report if it does not find that CA (otherwise, e.g. for CMP errors, the request will just be "rejected" without further clarification by the error message on your screen)

- The following files are created in your current directory. Mind, that "my-certificate-extra-0.crt" is only created, if you have in the CMP service option "Deliver CA in field:" the value set to "extraCerts". The file contains then the CA root cert and for each possible additional CA cert in a trust chain one extra file will be created. If the value is set to "caPubs", the CA-cert is delivered in the ca-Pubs field and discarded by the ssh-cmpclient (i.e. no file will be created for it).

```
[root@pki_sa ~]# ll
      8 -rw-------  1 root root      634 Oct 12 16:30 my-private-key.prv
      8 -rw-r--r--  1 root root      595 Oct 12 16:30 my-certificate-extra-0.crt
      8 -rw-r--r--  1 root root      536 Oct 12 16:30 my-certificate-0.crt
[root@pki_sa ~]#
```

- You can verify in the EE (see also 3.1) that the usecount is now decreased. If you don't limit by policy, the overcoming of the usecount will not prevent issuing more certificates!

3.2.2 Using ssh-cmpclient with external identity certificates

- The execution with external identity certificates requires that certificate- and key files are in place! They can be either created with PSK/Refnum or web enrollment, or a set of default certificates that is deployed along with the trial SW can be used. Naturally, the certificates must stand in a correct relation/hierarchy to each other (i.e. the trust anchor used in the command must be installed in the CA, see table with explanations)
- Run the program as follows to get a certificate based on external identity certificate, used parameters are described in the table below, output shown in the screenlog

```
[root@pki_sa Certs]# ssh-cmpclient INITIALIZE -P generate://pkcs8@rsa:1024/my-private-key
              -o my-certificate  -c /root/Certs/eNB_Factory_Cert.der  -k
              file://pkcs8@/eNB_Factory_Key.der -s 'CN=my-desired-subject-name'
              http://192.168.1.99:8080/pkix/ 'C=Br, O=CSI\=Sec-GS, CN=LTE basic CA'
```

```
Certificate =
  SubjectName = <CN=my-desired-subject-name>
  IssuerName = <C=Br, O=CSI\=Sec-GS, CN=LTE basic CA>
  SerialNumber= 11620
  SignatureAlgorithm = rsa-pkcs1-sha1
  Validity =
    NotBefore = 2011 Oct 12th, 20:18:09 GMT
    NotAfter  = 2011 Nov 11th, 20:18:09 GMT
  PublicKeyInfo =
    PublicKey =
      Algorithm = RSA
      Modulus n  (1024 bits) :
        15365237579181578753734142976610768398915712154760886063730933099659027
        06521256760439127228721849626264592130479512614909750230196886709728134
        91234980808225196438718258149368297886372875421440271583096683322015849
        02184285520423388747702589712987364336191359273533987871541529574490927
        9241390453757879155253931
      Exponent e (  17 bits) :
        65537
    Strength estimation as of July, 2000 considering NFS:
      Attack requires O(2^88) steps, which is roughly  equivalent to
      1.7 * 10^10 years of effort with 1GHz machine.
  Extensions =
    Available = authority key identifier, subject key identifier
    AuthorityKeyID =
      KeyID =
        47:9e:19:f2:30:8f:6d:56:6b:b5:ac:10:17:12:d6:fc:a5:d5:0d:2b
    SubjectKeyID =
      KeyId =
        04:34:1e:7d:55:ad:14:56:59:f3:5c:a2:e5:92:a3:5c:97:e1:5a:6b
  Fingerprints =
    MD5 = e0:d5:09:31:91:2d:e8:58:f5:c8:21:4d:30:ef:49:5f
    SHA-1 = d7:3e:c2:f1:fa:dd:c3:5d:44:48:9c:0b:e1:28:30:3b:20:70:49:d4
Do you accept the certificate above? y
Accepted user certificate; saving into file my-certificate-0.crt.
Fingerprints =
  MD5 = e0:d5:09:31:91:2d:e8:58:f5:c8:21:4d:30:ef:49:5f
  SHA-1 = d7:3e:c2:f1:fa:dd:c3:5d:44:48:9c:0b:e1:28:30:3b:20:70:49:d4

[root@pki_sa Certs]#
```

Parameter Name	Description
INITIALIZE	Triggers a CMP message of type IR
-P generate:…	Generates the private key file (locally) with given parameters (in example: 1024bit RSA key stored as PKCS#8 file ./my-private-key.prv in the current directory. The extension “prv” is added automatically by the system.
-o my-certificate-file	Will store the received certificate as “./my-certificate-file-0.crt” in the current working directory. The extension “-0.crt” is added automatically by the system.

-p 1949:passphrase	Is the Refnum (1949) and PSK (passphrase) connected to the EE for which the certificate is issued
-c ./eNB_Factory_Cert.der	The certificate file that will be used for authentication. The certificate of the issuing CA of this eNB_Factory_Cert.der must be installed as a trust anchor in the PKI, otherwise it wont work!
-k file://pkcs8@/eNB_Factory_Key.der	Private key file that belongs to above certificate. It is needed to sign the CMP messages for the certificate IR.
http://192.168.1.99:8080/pkix/	Is the URL to which the CMP service is configured in the PKI. The service must be connected to the CA!
'C=Br, O=CSI\=Sec-GS, CN=LTE basic CA'	Is the subjectName of the CA in the CA certificate. By this DN, the CA to issue the certificate is found. The ssh-cmpclient script will report if it does not find that CA (otherwise, e.g. for CMP errors, the request will just be “rejected” without further clarification by the error message on your screen)

❐ You can verify that the certificate has been really issued by the created files in the directory or by looking into the certifier DB using GUI.

3.3 Display and manipulate certs

With openssl suite, it is possible to do many certificate or TLS related tasks. Without touching the complete complexity and flexibility of the openssl CLI, few basic commands are introduced here and PKI users are encouraged to read the complete documentation of at least the CLI!

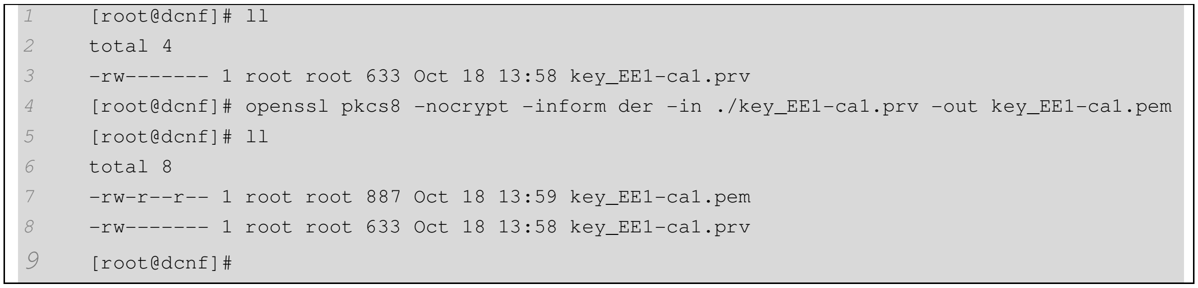
3.3.1 Display certificate

```
[root@pki_sa ~]# openssl x509 -inform der -in my-certificate-0.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 9916 (0x26bc)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=Br, O=CSI=Sec-GS, CN=LTE basic CA
        Validity
            Not Before: Oct 12 19:56:30 2011 GMT
            Not After : Nov 11 19:56:30 2011 GMT
        Subject: CN=my-desired-subject-name
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:c2:3c:3f:68:64:19:5f:91:73:38:3e:3a:32:70:
                    13:bb:f2:39:f7:17:ed:8a:a5:3d:cc:00:f3:39:93:
                    b7:59:89:ce:e5:fc:0a:1b:a7:c4:e3:b6:97:1f:36:
                    72:b8:21:3a:a5:b9:b8:4a:85:b4:55:a5:1a:17:14:
                    ab:43:a0:c0:3e:15:c7:93:d8:5b:9c:df:5a:28:ea:
                    30:02:df:72:37:63:28:7f:65:f7:42:57:95:ab:5e:
                    2a:ef:e1:df:e4:86:68:cf:4c:80:a8:04:0a:a7:8c:
                    26:0a:59:fe:78:a7:49:58:5a:55:a7:88:1c:ca:26:
                    fd:ee:b9:0c:54:fb:99:e3:ab
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:47:9E:19:F2:30:8F:6D:56:6B:B5:AC:10:17:12:D6:FC:A5:D5:0D:2B
            X509v3 Subject Key Identifier:
                E6:71:BC:FE:FA:7F:F9:A1:5E:64:37:0F:EA:68:3D:7A:C6:30:67:98
    Signature Algorithm: sha1WithRSAEncryption
        9e:95:56:86:8d:2b:db:ed:79:57:ff:67:bc:74:c4:ed:ef:5d:
        27:53:b2:b4:26:c2:00:af:39:1f:f0:e6:97:18:13:b1:f4:24:
        a7:72:ed:16:a1:92:d3:89:b9:33:af:b6:3f:8d:4b:ae:8e:e7:
        11:cf:53:a2:da:7c:27:18:1f:4e:46:c4:06:12:67:07:19:b1:
        32:d1:cf:90:8d:fd:a7:cb:e2:c5:04:da:22:e3:39:cb:a5:bc:
        71:80:38:82:be:d6:00:03:dd:14:14:b4:1f:cd:d5:83:84:e6:
        47:d8:c8:2a:50:f6:29:56:41:85:cd:1a:fe:dd:54:d9:72:9f:
        b9:2a
[root@pki_sa ~]#
```

MIND: If the certificate file is in PEM format, omit the "-inform" option!

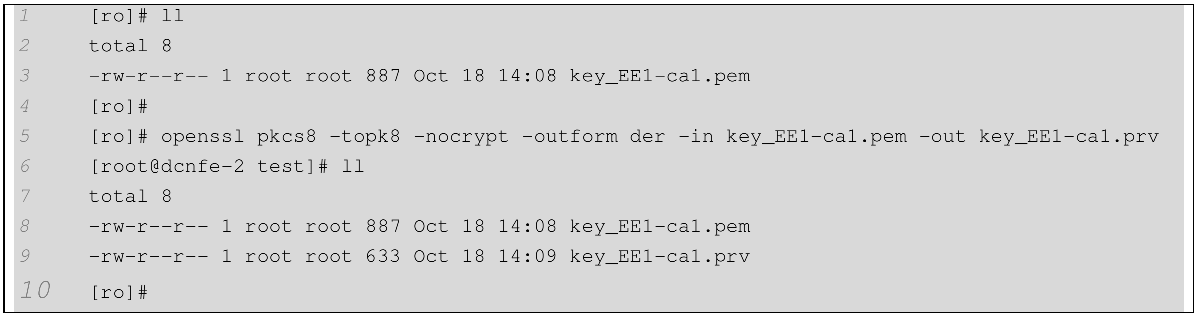
3.3.2 Convert keys between ber and pem

3.3.2.1 DER to PEM format



```
[root@dcnf]# ll
total 4
-rw------- 1 root root 633 Oct 18 13:58 key_EE1-ca1.prv
[root@dcnf]# openssl pkcs8 -nocrypt -inform der -in ./key_EE1-ca1.prv -out key_EE1-ca1.pem
[root@dcnf]# ll
total 8
-rw-r--r-- 1 root root 887 Oct 18 13:59 key_EE1-ca1.pem
-rw------- 1 root root 633 Oct 18 13:58 key_EE1-ca1.prv
[root@dcnf]#
```

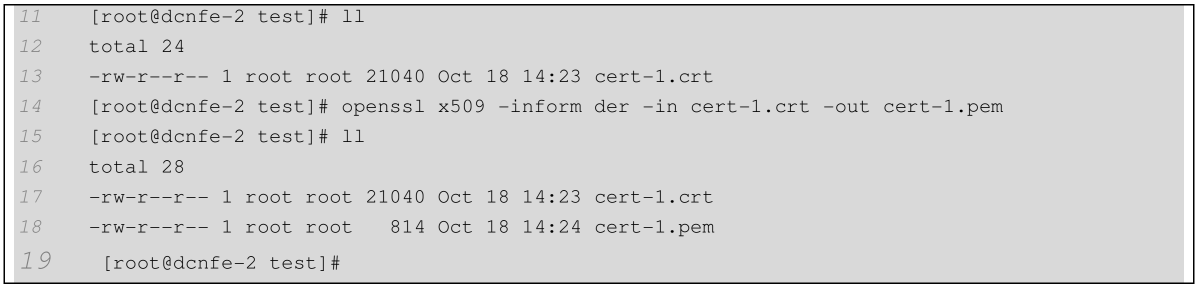
3.3.2.2 PEM to DER format



```
[ro]# ll
total 8
-rw-r--r-- 1 root root 887 Oct 18 14:08 key_EE1-ca1.pem
[ro]#
[ro]# openssl pkcs8 -topk8 -nocrypt -outform der -in key_EE1-ca1.pem -out key_EE1-ca1.prv
[root@dcnfe-2 test]# ll
total 8
-rw-r--r-- 1 root root 887 Oct 18 14:08 key_EE1-ca1.pem
-rw-r--r-- 1 root root 633 Oct 18 14:09 key_EE1-ca1.prv
[ro]#
```

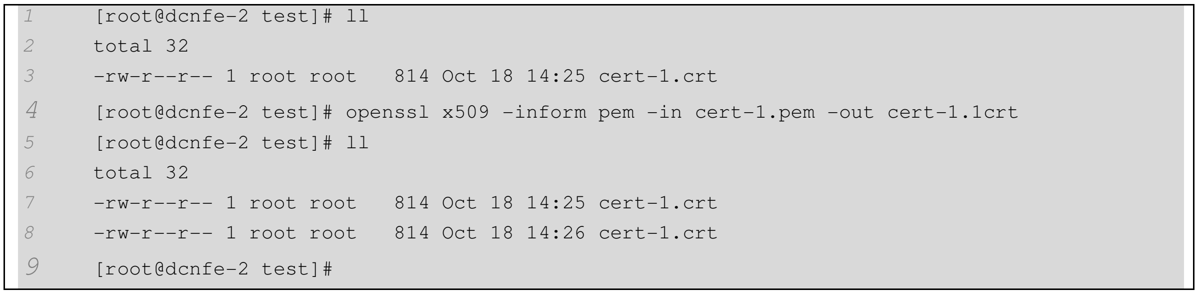
3.3.3 Convert certs between ber and pem

3.3.3.1 DER to PEM format



```
[root@dcnfe-2 test]# ll
total 24
-rw-r--r-- 1 root root 21040 Oct 18 14:23 cert-1.crt
[root@dcnfe-2 test]# openssl x509 -inform der -in cert-1.crt -out cert-1.pem
[root@dcnfe-2 test]# ll
total 28
-rw-r--r-- 1 root root 21040 Oct 18 14:23 cert-1.crt
-rw-r--r-- 1 root root   814 Oct 18 14:24 cert-1.pem
 [root@dcnfe-2 test]#
```

3.3.3.2 PEM to DER format



```
[root@dcnfe-2 test]# ll
total 32
-rw-r--r-- 1 root root   814 Oct 18 14:25 cert-1.crt
[root@dcnfe-2 test]# openssl x509 -inform pem -in cert-1.pem -out cert-1.1crt
[root@dcnfe-2 test]# ll
total 32
-rw-r--r-- 1 root root   814 Oct 18 14:25 cert-1.crt
-rw-r--r-- 1 root root   814 Oct 18 14:26 cert-1.crt
[root@dcnfe-2 test]#
```

4. Using LDAP with Certifier

LDAP servers are used in context of PKI and in NSN for 2 purposes:

- ❒ Delivery of an operator root certificate (see 4.1 below)
- ❒ As repository for BS/EE certs and CRLs (see 4.3 below)

Below gives an introduction on how to make sure your LDAP server is running properly, followed by straight guidelines to setup above use cases. Mind, that descriptions of this doc are general and only supposed to get the system running! There will be existing LDAP datamodels in an operator and DN naming-schemas which o project setup will have to follow. Those are not taken into consideration here!

4.1 Ensure LDAP server functionality

All descriptions here focus on openldap running on linux. To prepare an LDAP server to provide the operator root certificate, do the following:

- ❒ Check by the following command, that the LDAP objectClass pkiCA is supported in any of your schemas.

```
[root@giovanni ~]# cd /etc/openldap/schema [root@giovanni schema]# grep "pkiCA" *
core.schema:objectclass ( 2.5.6.22 NAME 'pkiCA'
[root@giovanni schema]#
```

- ❒ Check that openldap is running and listening on port 389 (see also 2.1.2.5 how to make sure port TCP 389 is not blocked by the iptables firewall)

```
[root@pki_sa ~]# netstat -tunap |grep 389
Tcp   0   0.0.0.0:389      0.0.0.0:*     LISTEN     15553/slapd
[root@pki_sa ~]#
```

4.2 Provide a CA root cert in LDAP

In older version of BS software, the delivery of a operator CA cert is not expected in either caPubs (RU30/RL30) nor extraCerts (RU40/RL40) field of the CMP response. Instead, the BS fetches the CA cert from an LDAP server. The CA cert can be manually planted into theis LDAP server as in the procedure described in

Check that the LDAP server is running

- ❒ Create the following ldif datastructure to prepare the LDAP server in a later step (save this as ldif file. Use as “`cACertificate;binary::`” the PEM format of the actual operator CA certificate. Mind, that between the first character of the cert and the “binary::” you need a blankspace! Mind also, that the entire PEM certificate shold be added as one line! Below, lines 25-36 are to be added to the ldif file as ONE line without linebreaks within the certificate!)

```
# RU20CA-cert
dn: dc=RU20CA-cert
objectClass: organization
objectClass: dcObject
o: NSN
dc: RU20CA-cert

# Root, RU20CA-cert
dn: ou=Root,dc=RU20CA-cert
objectClass: organizationalUnit
objectClass: top
ou: Root

# CertificateAuthorityRoot, Root, RU20CA-cert
dn: ou=CertificateAuthorityRoot,ou=Root,dc=RU20CA-cert
objectClass: organizationalUnit
ou: CertificateAuthorityRoot

# RU20TestCA, CertificateAuthorityRoot, Root, RU20CA-cert
dn: cn=RU20TestCA,ou=CertificateAuthorityRoot,ou=Root,dc=RU20CA-cert
objectClass: top
objectClass: pkiCA
objectClass: device
cn: RU20TestCA
cACertificate;binary:: MIICfDCCAeWgAwIBAgICJTcwDQYJKoZIhvcNAQEFBQAwUTELMAkGA1
EBhMCQnIxDDAKBgNVBAoTA05TTjEQMA4GA1UECxMHQ1NJLVNFQzELMAkGA1UECxMCR1MxFTATBgNV
BAMTDExURSBCYXNpYyBDQTAeFw0xMTEwMDMxODIxNDRaFw0xMjEwMDIxODIxNDRaMFExCzAJBgNVB
AYTAkJyMQwwCgYDVQQKEwNOU04xEDAOBgNVBAsTB0NTSS1TRUMxCzAJBgNVBAsTAkdTMRUwEwYDVQ
QDEwxMVEUgQmFzaWMgQ0EwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJTpsPmZWDXAcGxFvVs
P1RStsU8SpwOs3qAuM60B7uovuO1Rl30tbDJYHs2I3puqW5UslOKrJhV3B6BJl+ynABx4DAeXKAPG
R0N44InG4Eommt2kkX3QUOb8/e3YZLWSS045UaOkRvSSRK89QG2mXlq4/Ha2QNM1pp3ejs/KjYvODA
gMBAAGjYzBhMB8GA1UdIwQYMBaAFEqDEgKrATPu6LS9/QN/JyK9K/reMB0GA1UdDgQWBBRKgxICqw
Ez7ui0vf0DfycivSv63jAOBgNVHQ8BAf8EBAMCAf4wDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0
BAQUFAAOBgQBz3EZ58zSXRHOvjA4AeJ3LJGtls057NmPrMpaQjNGdie2JbFtm19OFHdQXEst+cFYe
LSp5uTUDiiSJCiBg1YKINYMMHmuud3PcBEj3AlmFD8HL8CkbM4jt3Myp574tH1/pugNPyxTUpQ3FM
r3pE9bCW2ABnQyJPj6HAXAe8w8v5w==
```

❒ Add the following entry to your /etc/openldap/slapd.conf

```
#... inside /etc/openldap/slapd.conf :
database bdb
suffix       "dc=RU20CA-cert"
rootdn       "cn=manager,dc=RU20CA-cert"
rootpw       password
directory    /var/ldap/ru20ca
mode 0600
```

❐ make sure to have a “clean” LDAP directory by creating the datafiles directory in Linux

```
[root@pki_sa ~]# rm -rf /var/ldap/ru20ca
[root@pki_sa ~]# mkdir /var/ldap/ru20ca
[root@pki_sa ~]# chmode 0600 /var/ldap/ru20ca
[root@pki_sa ~]#
```

❐ Restart ldap

```
[root@pki_sa ~]# /etc/init.d/ldap restart
Stopping slapd                  [  OK  ]
Starting slapd                  [  OK  ]
[root@pki_sa ~]#
```

❐ Add the ldif file

```
[root@dcnfe-2 ~]# ldapadd -h 192.168.1.69 -x -w password -a -f default.ldif  -D
"cn=manager,dc=RU20CA-cert"
adding new entry "dc=RU20CA-cert"

adding new entry "ou=Root,dc=RU20CA-cert"

adding new entry "ou=CertificateAuthorityRoot,ou=Root,dc=RU20CA-cert"

adding new entry "cn=RU20TestCA,ou=CertificateAuthorityRoot,ou=Root,dc=RU20CA-cert"

[root@dcnfe-2 ~]#
```

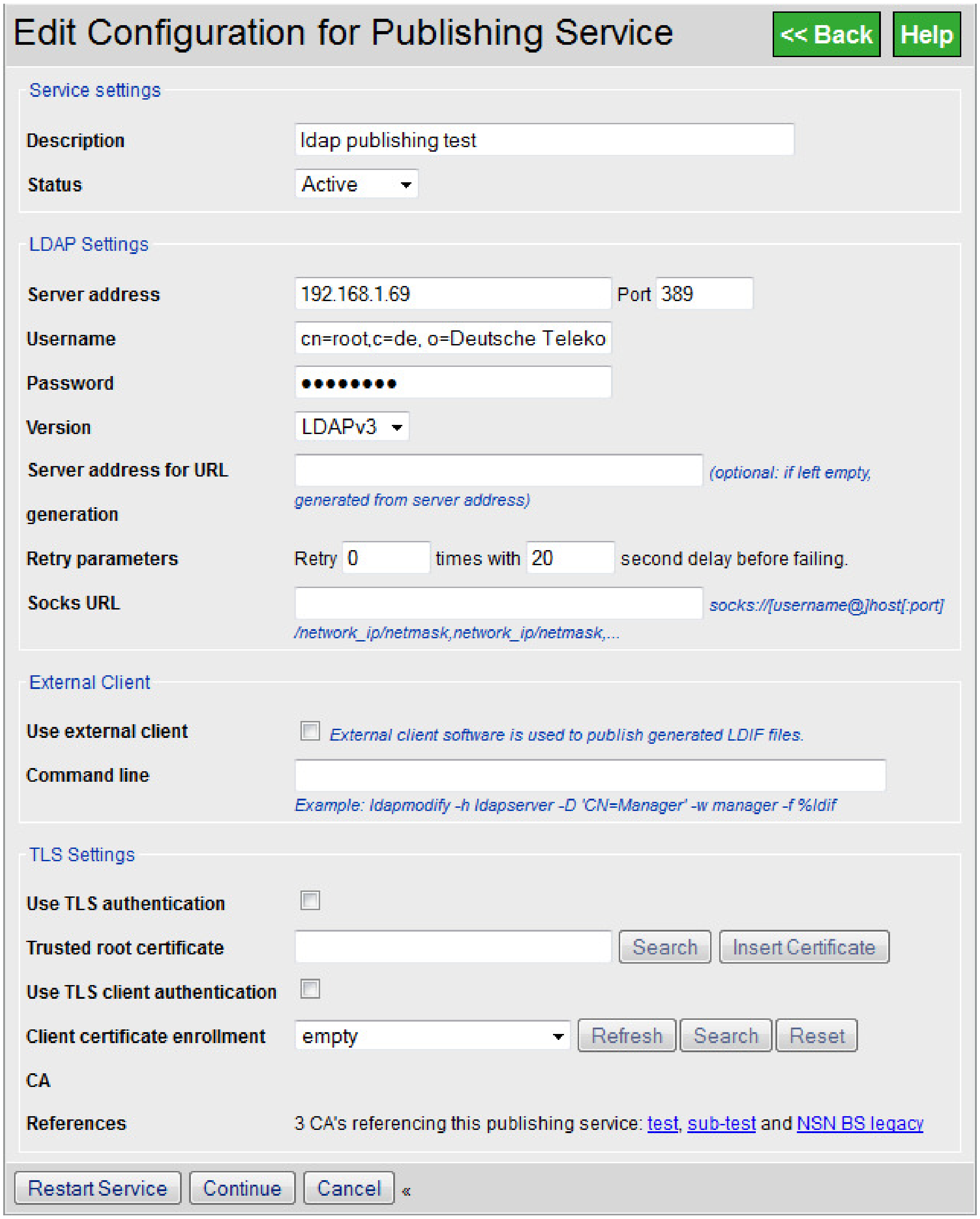
4.3 Publishing into LDAP with Insta

Publishing certificates or CRLs into LDAP requires 2 steps in the certifier: First, the publishing service must be configured to connect to a matching datastructure of an existing LDAP server. Second, the certificates and/or CRL must be published into this directory

4.3.1 Create a publishing service

Publishing services define the ldap-client setting needed to send bind-requests to the server under which root DN (suffix in slapd.conf) the certificates will be held. You can secure LDAP connections with certificates, which is highly recommendable. This document will not describe how to set up LDAP over TLS, but how to get a basic connection.

- ❒ Similar to what has been done in Chapter 2.3 for CMP, do now add a Service of type “LDAP publishing”. Under Servers, in the “services” pulldown menu choose “Publishing Service” and press the “Add Service” button. You will get to the “Edit Configuration for Publishing Service” menu. You need to fill in at least the following parameters:



Service settings

Description: ldap publishing test

Status: Active

LDAP Settings

Server address: 192.168.1.69 Port 389

Username: cn=root,c=de, o=Deutsche Teleko

Password: ●●●●●●●●

Version: LDAPv3

Server address for URL generation: (optional: if left empty, generated from server address)

Retry parameters: Retry 0 times with 20 second delay before failing.

Socks URL: socks://[username@]host[:port]/network_ip/netmask,network_ip/netmask,...

External Client

Use external client: ☐ External client software is used to publish generated LDIF files.

Command line: Example: ldapmodify -h ldapserver -D 'CN=Manager' -w manager -f %ldif

TLS Settings

Use TLS authentication: ☐

Trusted root certificate: Search Insert Certificate

Use TLS client authentication: ☐

Client certificate enrollment CA: empty Refresh Search Reset

References: 3 CA's referencing this publishing service: test, sub-test and NSN BS legacy

Figure 18: *Basic configuration for a minimum publishing service.*

Parameter Name	Description
Description	Any descriptive string. The publishing service will be shown under this description in the CA configurations where you can refer to it.
Status	Must be "Active" You can temporarily deactivate services here, e.g. if you are troubleshooting or doing LDAP tree configuration and bind-attempts might disturb you wireshark outputs.
Server Address	Use the IP address to which your LDAP server is listening, see also 4.1 and the second command, where netstat shows you this IP address and port. You should try to avoid changing the port, you can have secure LDAP (via START TLS directive) also on port 389.
Username	Is normally the rootdn from /etc/openldap/slapd.conf if you have a rootdn defined. If not, it can be any username/password combination that has access to the server and can write to the branches of the tree (DN's) to which you want to write. Mind, that LDAP usernames are given in form of DNs as well and not as plain name like "root", you are referoing to users inside the LDAP tree! See also LDAP ACL documentation
Password	Password (or userPassword attribute) of the Username.
Version	Always use version 3
Client certificate Enrollment	This normally shows any of the CAs on your Insta installation, in the example screenshot the CA is called "empty". The field has no meaning as long as you do not use secure LDAP. If you use secure LDAP, the CA that is given here must be the CA from which clients authenticate. This would replace the need for username and password and instead switch to client-authentication. Mind, that this requires also settings on Linux level to place the proper certificates on client (Insta) and LDAP server side as well as suitable authentication configuration.

❒ Press "continue" and in the next screen "commit changes" after you have entered parameters to get a screen similar to that in *Figure 18*. Check in the "Edit Server Entity" screen later, if the publishing service is really running with the "Status Query" button.

❒ MIND: In order to have the publishing service to function, your Username and password must match and the baseDN (suffix in /etc/openldap/slapd.conf) must provide the datastructure you refer here! Further, you must have write-access to the actual certificate or CRL branch of the tree, which can be a totally different DN and is defined in the actual distribution point in the CA configuration!

4.3.2 Define publishing settings for EE certificates

In order to publish certificates for EEs (namely: Base stations or Security Gateways) into LDAP, you have to edit the CA publishing settings for Certificates. After completing the configuration as shown below, you will have the following result:

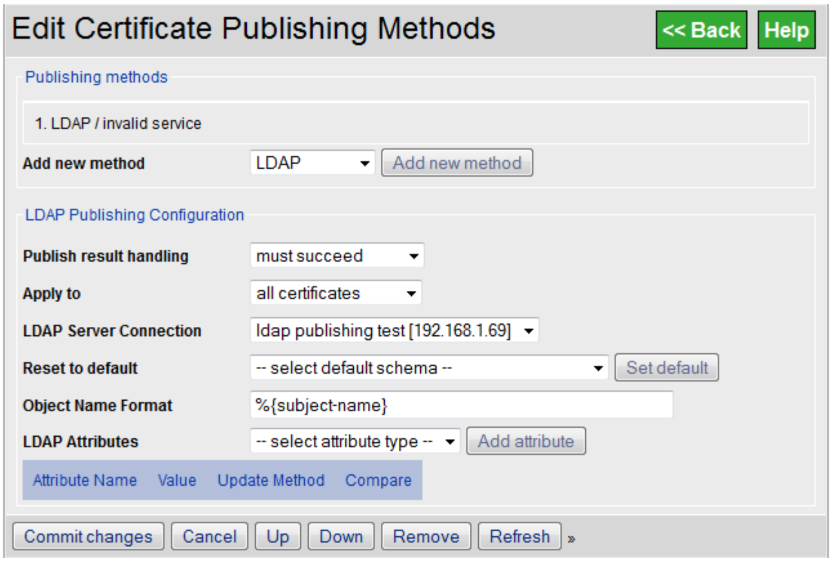
- Each time a new certificate is issued from that CA, an LDAP bind will be sent to the DN you refer in the Publishing Service.

- After successful bind, an ldapsearch will be made for the certificate holder, whereby the search parameters are according to what you defined in the “Object Name Format” of the Certificate Publishing Method (this parameter has of course a variable part that will be specific to the subject of the certificate)

- If above search is successful, the entry will be modified according to your Publishing Method Attributes. If the search is not successful, the entry will be created since it has not existed before.

□ Similar as in Chapter 2.2, open the “Certification Authority” screen of the existing CA for which you want to activate publishing. In the “Certificate Settings” section of the screen, press the “Edit Publishing Methods” button..

□ If no publishing methods have been defined so far, you can now only choose between “LDAP” or “External”. Choose LDAP and press the “Add new Method” button. This will bring you to the “Edit Certificate Publishing Methods” Screen as depicted below.



Edit Certificate Publishing Methods

Publishing methods

1. LDAP / invalid service

Add new method: LDAP

LDAP Publishing Configuration

Publish result handling: must succeed

Apply to: all certificates

LDAP Server Connection: ldap publishing test [192.168.1.69]

Reset to default: -- select default schema --

Object Name Format: %{subject-name}

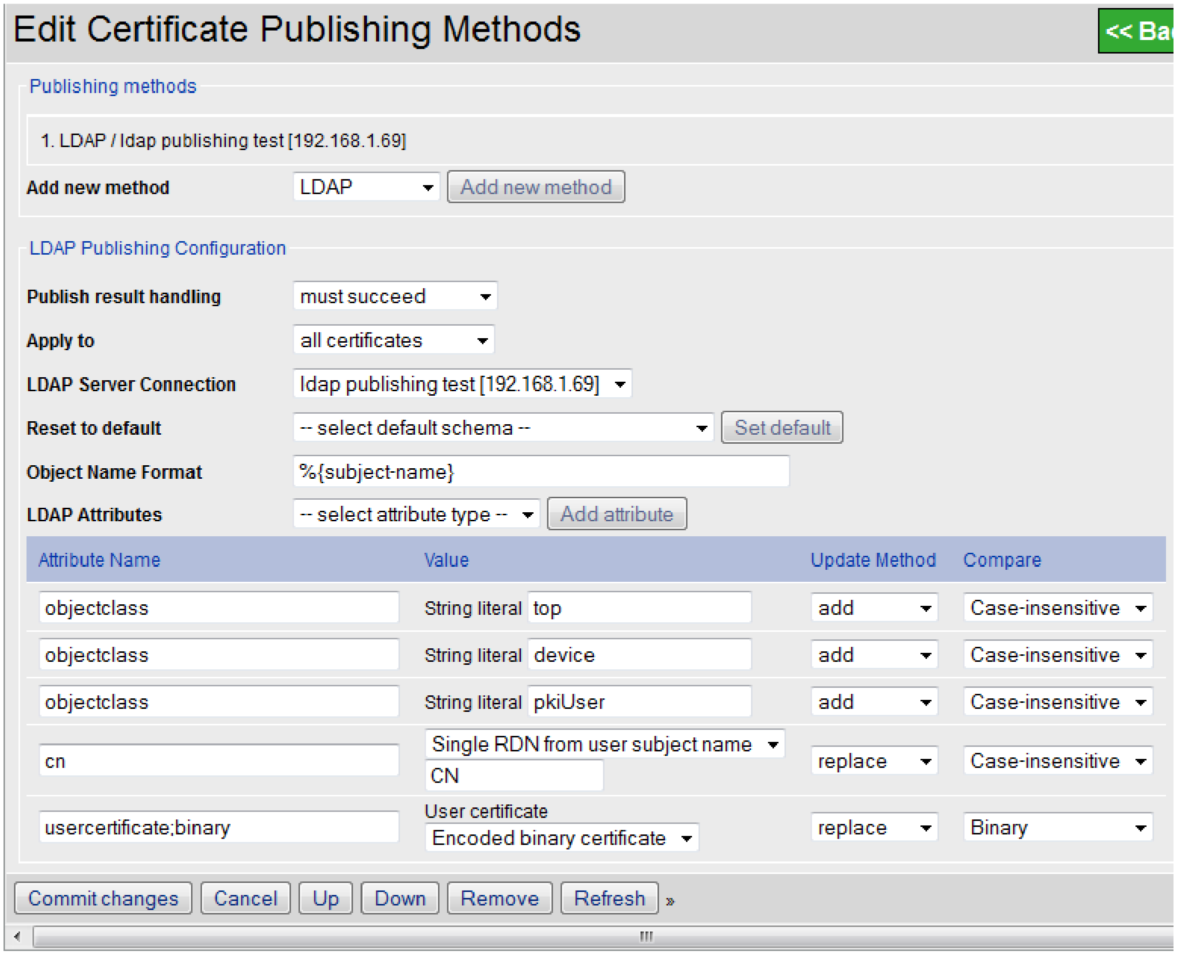
LDAP Attributes: -- select attribute type --

Attribute Name	Value	Update Method	Compare

Figure 19: *“Fresh” (i.e. no previously existing) Publishing Methods definition.*

Parameter Name	Description
Publish result handling	Defines how certifier handles the exit staus of the publishing operation. In case of “must succeed” the “retries” as configured in the publishing service will be applied and an SNMP trap will be fired for each unsuccessful attempt.
Apply to	Scope of the publishing. In our context, apply to all certificates.
LDAP server connection	Allows you to chose from the (LDAP) Publishing services you have defined under 4.3.1.

❒ If you already had a publishing method defined before – regardless if it was a working or not working one – you will see the screen as depicted below, where already Attributes for the LDAP modify or add statements are defined. If you see no attributes, select "LDAPv3 pkiUser schema" under reset to default/--select default schema --. And press the "set default" button. This step defines the LDAP schema by which publishing works. Make sure that schema is supported in your LDAP server installation (see also 4.1 step 1). Setting a schema will expand the screen and display Attribute definitions.



Edit Certificate Publishing Methods

Publishing methods

1. LDAP / ldap publishing test [192.168.1.69]

Add new method LDAP

LDAP Publishing Configuration

Publish result handling must succeed

Apply to all certificates

LDAP Server Connection ldap publishing test [192.168.1.69]

Reset to default -- select default schema --

Object Name Format %{subject-name}

LDAP Attributes -- select attribute type --

Attribute Name	Value	Update Method	Compare
objectclass	String literal top	add	Case-insensitive
objectclass	String literal device	add	Case-insensitive
objectclass	String literal pkiUser	add	Case-insensitive
cn	Single RDN from user subject name CN	replace	Case-insensitive
usercertificate;binary	User certificate Encoded binary certificate	replace	Binary

Figure 20: *Publishing Methods definition after selecting the default schema (attributes are added).*

❒ The Attribute definitions of the default schema are sufficient and represent the information that will be placed into your LDAP directory. In the above Figure, this is: The CN of the certificate subject and the PEM encoded certificate itself, as binary object. Those Attributes are arguments / search filter to the LDAP operations of the publishing service. In order to have this operation succeed, you must as a last step edit the "Object Name Format" such, that it matches the absolute DN under which your certificate will be found in the LDAP tree. Normally, this is

- The chain of RDNs in the LDAP tree up to the level where certificates are inserted, e.g.

  o=operator, c=de, ou=RAN BTS CA

- plus the representation of the certificate subject name, which will be taken from the certificate itself. In this example, the “cn” value from the usersubjectname attribute is taken, represented by, e.g.:

  cn=%{subject-name:cn}

- Both values have to be chained to comprise the actual CN under which the certificate will be found, separate the values by colon “,” :

  o=operator, c=de, ou=RAN BTS CA, cn=%{subject-name:cn}

❒ After completing those steps, you will be able to see the newly issued certificates in the ldap tree. For example with

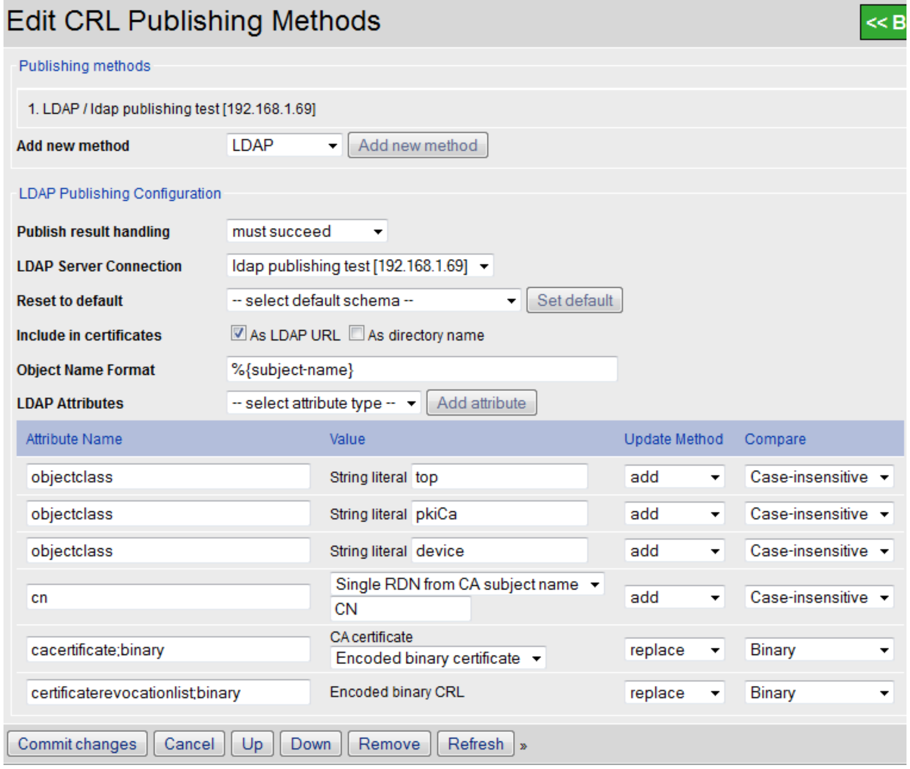
```
ldapsearch -x -w [password] -D [rootDN] -h [ldapserver] -b [cert-DN]
```

Parameter Name	Description
[password]	Is normally the user-password you defined in Chapter 4.1 for this username
[rootDN]	Is normally the username you have defined in 4.1 and who has access to that branch of the LDAP tree
[ldapserver]	Is the IP address of the LDAP server as defined in 4.3.1.
Cert-DN	Is the DN under which the certificate is stored, where “cn=” (from above example) has turned into the subject name used in the certificate (or at least the cn part of it) as in o=operator, c=de, ou=RAN BTS CA, cn=BTS-NSN-1235432467

4.3.3 Define publishing settings for CRL

Configuration of CRL publishing is similar than for EE certificates, only testing purposes and location in the LDAP directory are different. It is described here only for completion.

- ❒ Similar as in 4.3.2 / Figure 19, the CA configuration has a section for CRL publishing. Press the “Edit publish Settings” button of the CA configuration. Then follow the same instructions as for EE certificates and set the default schema to “LDAPv3 pkiCA” to get into the below screen:



Edit CRL Publishing Methods

Publishing methods

1. LDAP / ldap publishing test [192.168.1.69]

Add new method: LDAP ▾ Add new method

LDAP Publishing Configuration

Publish result handling: must succeed ▾

LDAP Server Connection: ldap publishing test [192.168.1.69] ▾

Reset to default: -- select default schema -- ▾ Set default

Include in certificates: ☑ As LDAP URL ☐ As directory name

Object Name Format: %{subject-name}

LDAP Attributes: -- select attribute type -- ▾ Add attribute

Attribute Name	Value	Update Method	Compare
objectclass	String literal top	add ▾	Case-insensitive ▾
objectclass	String literal pkiCa	add ▾	Case-insensitive ▾
objectclass	String literal device	add ▾	Case-insensitive ▾
cn	Single RDN from CA subject name ▾ CN	add ▾	Case-insensitive ▾
cacertificate;binary	CA certificate Encoded binary certificate ▾	replace ▾	Binary ▾
certificaterevocationlist;binary	Encoded binary CRL	replace ▾	Binary ▾

Figure 21: *Default screen for definition of CRL publishing methods with pkiCA schema as default.*

- ❒ Similar as in the EE publishing, the basic settings should be adjusted and the ObjectNameFormat might be different. It has to match the DN of the CRL in the LDAP directory, which is maybe a different “higher tree”/different branch than the EE certificates. It should also have different write-permissions and hence a different publishing service might be needed (one with write access to CRLs and not only to CA or EE certificates).
- ❒ MIND: That also for CRL, all datastructures you use must exist in LDAP and the chance that you get it working without knowing how to handle LDAP data trees and LDIF files is very low! This document only describes the INSTA part!

5. Security Options

In HAPF2.0 onwards, NSN delivers a pre-hardened RedHat OS configuration for Insta Certifier. However, there are some recommendations on how Insta services – protocol staxcks and backend functionality – should be configured to gain secure operation on application level.

Mind the following:

- After installation, a root operator admin is created with a password "admin". The password must be changed. It is also possible to delete the admin operator after creation of a new operator with same access level. This is recommended if operators have personal user accounts. Mind, that features like “dual admin” can significantly increase the security level of your system and at the same time only make sense if each admin uses an individual account.
- If above recommendation is followed and the default admin account is changed or deleted, the daily-backup.sh script will not be able any more to create online database backups! Verify the NSN HAPF2.0 admin documentation to understand how to configure secured access for scheduled backups in this case. Mind, that in every case you will require to store credentials in any one form on the server if you want to run unattended jobs. It is therefore a question of policy, if storing of credentials with root-only access can be allowed or if such storing in general is prohibited (by policy) and only interactive authentication allows access to privileged information!
- Engine TLS private key should be protected with a master password. This can be done by setting "Encrypt TLS private key" in System Parameters page. The Master password should also be changed from its default value.
- If possible, the Administration service should be bound to the internal TLS IP OAM of the Backends only (VLAN1 virtual IP), and not be configured on the Frontends along with the other services. Protect the service with TLS and preferably with client authentication.
- The default Web Enrollment service should be removed if it is not required; for example when using CMP, SCEP or off-line certification via Administration GUI. If Web Enrollment service is used it should be protected with TLS.
- All Certificate services (CMP, SCEP, Web Enrollment) should be bound only to the required IP of the FE bond0:${VLANID[X]} interface by explicitly specifying the corresponding IP address on the FE VLANs
- In a multi-CA environment it is recommended to use different services for accessing CAs from different organizations; for example a separate CMP service bound to different ports (e.g. LTE or 3G CMP)
- DoS settings (dos (host-rate-limit <n>)) in server.conf should be checked fordifferent services. The rate limit value specifies how many requests from a single host is allowed in 10 seconds. However, this does not prevent DoS attacks against OS's TCP/IP stack implementation. For this, the RedHat kernel settings and iptables provide separate protection.

❒ Operator access levels should be explicitly set for needed CAs and should only have level sufficient for operator's purpose. Only admin or similar operator should have "super-user" access to "All CAs".

❒ Access to CAs for delegated RAs should be explicitly specified and never allow access for implicit "All CAs".

❒ Accessible CAs for certain services (Web Enrollment, CMP, SCEP, Validation Authority) should be set explicitly and always use "Only selected CAs" mode instead of implicit "All CAs" or "All except selected CAs".

❒ Publishing service (to LDAP server) must be protected with TLS, refer to the HAPF2.0 admin note “set up LDAP services on the FE”