



UNDERSTAND, INSTALL AND DEPLOY AN EJBCA

Three deployment scenarios for the EJBCA
Certification Authority

Abstract

This training document describes how to set-up EJBCA with JBOSS 7 in three scenarios: a) issuing self-signed certificates; b) issuing SSL certificates through a simple CA with certificate revocation services (OCSP and CRL); and c) setting up a more complex CA with: two subordinated CAs, user self-registration and an advanced authorization schema.

Adrià Massanet
adria.massanet@gmail.com

Contents

Scope & audience.....	3
Fast review on PKI.....	4
The PKIX architecture model.....	5
PKIX components	5
End Entity	5
CA (Certification Authority).....	5
RA (Registration Authority)	6
Repositories.....	6
PKIX management functions	6
Registration	6
Initialization.....	6
Certification.....	6
Key pair recovery.....	7
Key pair update	7
Revocation request	7
Cross-certification	7
Additional management functions.....	7
PKIX management protocols	7
PKIX certificate discovery and validation protocols	8
EJBCA 10m Bird's-eye view	9
Major functionality.....	9
Operation web portals	9
Private administration portal	11
Public administration portal.....	12
System operation	13
Integration with other components.....	13
Supported software stack	14
Installing EJBCA and issuing self-signed certificates.....	15
Installation.....	15
Creation of a certificate template	17
Creation and exportation of a certificate	18
Basic Certification Authority	21
Creation of the EDPSCA ROOT Certification Authority.....	22
Creation of the certificate and End Entity profiles.....	23
Create the SSL client profile	23

SSL Server certificate profile	25
SSL Client entity profile	25
SSL Server entity profile	26
Issuing the certificates.....	26
Creating end-users	26
Generating a client SSL certificate for john	27
Generating a client SSL certificate for peter	28
Generating the SSL server certificate	28
Revoking john's certificate and generating a new CRL.....	30
Advanced Certification Authority.....	31
Create the administration and NQC subordinated CAs.....	31
Set-up EJBCA to use SSL certificates from EDPSADMIN SUBCA	33
Create a Super Administrator certificate and set permissions.....	33
Create a new JBOSS SSL server certificate and replace the existing one	34
Revoke the old management CA.....	36
Create EDPSNQC CA administrators.....	36
Create NQC certificate templates and profiles	37
Configure the website to enable self-registering to the portal and to send emails.....	40
Creating a self-registered 1.3.130.2.1 user	40
Registering the users as RA administrators.....	43
Configuring JBOSS to accept the RA administrator certificates as valid SSL client certificates.....	43
Creating a 1.3.130.2.2 user with dual control (two approvals)	44
Creating an EJBCA service to automatically update CRLs.....	45
Publishing issued certificates in an LDAP directory.....	46
Install and configure OpenDS.....	46
Configure EJBCA to publish certificates in the LDAP	47
Republishing all certificates in the LDAP	49
Accessing the CA remotely.....	49

Scope & audience

This document is designed for people with basic knowledge of Public Key Infrastructure and X509v3 Certificate environments, and is structured to provide a practical idea on how to operate an EJBCA Certification Authority in a progressive way:

- **Fast review on PKI** refreshes concepts about what Public Key Infrastructure is
- **EJBCA 10m Bird's-eye view** gives a general idea about the EJBCA software
- **Installing EJBCA and issuing self-signed certificates** describes how to install a test EJBCA installation and issue self-signed certificates
- **Basic Certification Authority** explains how to operate a basic Certification Authority to issue client and server SSL certificates
- **Advanced Certification Authority** describes how to develop a more advanced certification authority using more features of the EJBCA software

JBOSS Application Server 7.1.1, EJBCA Community Edition 6.0.3, and the H2 database engine are used. We prefer using this configuration as the target for this document is training and not how to deploy EJBCA in production or high availability environments (this would require a specialised knowledge outside the EJBCA software scope). Moreover, it eases making full backups and restoring them if something becomes wrong.

This document is not intended to be an EJBCA guide. You can find a complete EJBCA guide and HOWTOs available in the <http://www.ejbcna.org> site.

Fast review on PKI

Information security is crucial for any organization that deals with sensitive data. Any data transaction should fulfil the following security needs:

- authentication of the identity of the entity with which the information exchange is being conducted;
- confidentiality and confidence in the integrity of the information exchanged;
- establishment of non-repudiation agreements; and
- digitally notarising and securely time stamping.

PKI (initials for Public Key Infrastructure) is a long term solution based on cryptography that delivers these services efficiently and in a cohesive manner. Cryptography is aimed at securing communication in the presence of third parties by using keys to encrypt and decrypt data. Encryption converts text in unintelligible through a mathematical formula; while decryption reverts the process.

There are two types of cryptography: secret key or symmetric and public key or asymmetric. Symmetric key cryptography uses the same key to encrypt and decrypt data. It was the only form of cryptography used until the mid 1970s. Although efficient, it had two problems: it could not support certain security services and the secret keys had to be distributed securely to the communicating parties.

Public key cryptography uses key pairs: the private key, which must be kept secret, and usually under the owner control; and the public key, which can be distributed to anyone that wants to exchange data with the person holding the private key. Although the keys in a pair are mathematically related, the private key cannot be determined by knowing the public key.

Asymmetric cryptography offers two advantages when compared to the symmetric one: facilitating key delivery and reducing the number of keys that must be used within a system. In public key cryptography each person needs just a key pair, whereas in secret key cryptography each person requires a different key for each person with whom he communicates.

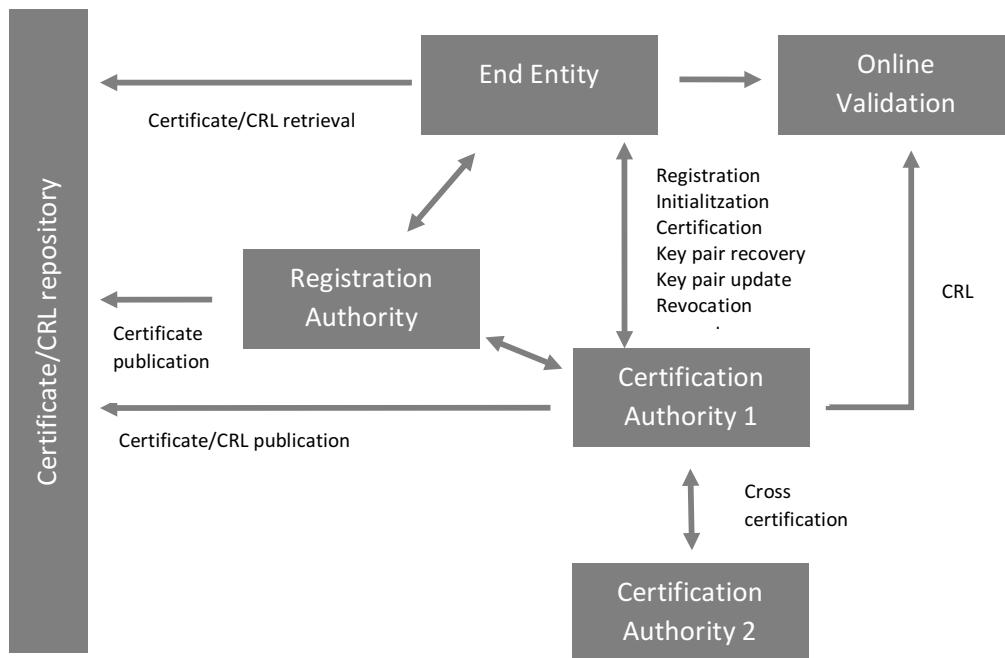
How is trust granted in a Public Key Infrastructure? By using a data element called *digital certificate* or *public key certificate* which binds a public key to identifying information about its owner. Digital certificates allow people to digitally sign a document; thus, verification is made available immediately if the relying party determines that the risk inherent in not doing so is significant enough. The trust is built into the infrastructure by design.

PKI is an authentication technology that integrates public key cryptography for digital signing and key management, and symmetric key cryptography for encrypting. Through scalable key and certificate life cycle management PKI also provides data confidentiality and data integrity. The X.509 standard defines its framework.

The PKIX architecture model

Under this section, a definition of each PKIX component is described, together with how it relates to each other component.

PKIX components



End Entity

Although usually end-users, End Entities can also refer to devices (e.g. servers or routers) or to any other end entity defined in the subject name of a public key certificate. End Entities can also consume or support PKI-related services (e.g. a RA is considered an End Entity from the point of view of a CA).

To participate in a PKI, servers and end-users (that is, End Entities bound to certificates) must first enlist in it.

CA (Certification Authority)

Public keys are distributed in the form of certificates and the CA is the only component infrastructure that can issue them.

CAs also issue Certificate Revocation Lists (CRLs). A CRL is a black list of certificates: it enumerates revoked certificates along with the reason(s) for revocation.

Although a CA may also support administrative functions, these tasks are often delegated to one or more Registration Authorities.

There are two types of CAs: root CAs and intermediate or subordinate CAs. While root CAs are trust anchors, which means that their trust is assumed not derived, intermediate CAs must be

verifiable and verified by following their certification path (chain of certificates between a certificate and its trust anchor CA) in order to be sure of their trustworthiness.

RA (Registration Authority)

The RA is an optional component that can assume CA administrative functions, such as:

- End entity registration process (including verification of the End Entity identity)
- Validating the subject attributes of the certificate requester
- Verifying that the subject holds the private key that is being registered
- Generation of shared secrets to support the initialization and certification process
- Public/private key pair-generation
- Intermediary between the CA or CAs and the End Entity
- Parameter validation of public keys presented for registration

One of RAs main advantage is to reduce overall costs for large and geographically dispersed organizations that require their users to register physically. The offloading of administrative functions from the CA also allows operating it offline, which reduces the risk of suffering remote attacks.

Repositories

Any method used for storing certificates and CRLs so that they can be retrieved by End Entities . Usually LDAP Directory Servers are used.

PKIX management functions

A Public Key Infrastructure needs to support the management functions detailed below.

Registration

Before End Entities can interact with a PKI, they must first enlist into the infrastructure. This is done through registration, initialization and certification, which can be managed either as separate functions or combined.

By registering, the End Entity makes itself known to a CA, which verifies its identity.

Initialization

Initializing the associated trust anchor with the End Entity follows registration. In this step, the public and private key pair related to the End Entity is also created. Key pairs can be generated by the End Entity client system, or by an RA, CA or some other component such as a hardware security module.

Certification

The enlistment process ends with the CA issuance of the End Entity public key certificate. If the key pair is generated external to the CA, the public key component must be transmitted to the CA in a secure manner. Once generated, the certificate is returned to the End Entity and/or published to a certificate repository.

Key pair recovery

Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both. It is solely to recover their encryption/decryption use that key pair recovery has been conceived.

For example, key pair recovery allows End Entities to restore their encryption/decryption key pair from an authorized key backup facility in case of forgotten passwords, corrupted disk drives, etc.

Key pair update

Update of a key pair may be needed in two occasions: when the lifetime of a certificate expires or when a certificate is revoked. By updating a key pair, a new key pair is generated and a new public key certificate is issued.

Revocation request

Sometimes a certificate has to be revoked before it expires (e.g. name change or change in affiliation). After an End Entity or RA request certificate revocation to the CA that issued that certificate, either the CRL Issuer or the CA itself must make certificate revocation information available. X.509 defines a method for publishing this information via Certificate Revocation Lists (CRLs.)

It is important for End Entities to check the revocation status of all certificates in a given certification path to ensure that data is protected.

Cross-certification

Cross certification enables entities in one PKI to trust entities in another PKI. In a hierarchical trust model with a root CA and subordinate CAs, the root CA can cross-certificate the subordinate CAs but not the other way round.

Additional management functions

Some management functions may be required just in certain environments. These include:

- CA key update announcement;
- certificate announcement: method to announce the existence of a certificate when no other method is available;
- revocation announcement;
- CRL announcement;
- certificate confirmation: used by an End Entity to accept or reject the issued certificate;
- key archive: used to request private decryption key backup.

PKIX management protocols

Management protocols support online protocol exchanges between various PKI components. Certificate Management Protocols (CMP) are the most comprehensive and flexible PKI management protocols, as they can accommodate various technical, operational and business models.

PKIX certificate discovery and validation protocols

The need to offload portions or all of the certificate discovery and/or validation process from the client system has driven to the creation of developing protocols that assume these tasks.

OCSP is the predecessor for these protocols. It returns real-time responses to client queries, and provides a method for returning certificate status on demand. OCSP functionality is however limited, that's why other protocols have been created.

SCVP allows offloading the certification path and/or validation process from the client's system.

EJBCA 10m Bird's-eye view

EJBCA is a complex piece of software. In this chapter we see the characteristics of EJBCA in a summarized way:

- **Major functionality** describes the main characteristics supported by the Open Source and the Enterprise Edition.
- **Operation Web Portals** details the functionalities shown in the main menu for the private administration portal and the public portal.
- **Systems Operation** explains the available options for system operations.
- **Integration with other components** lists the possible connections with other systems.
- **Supported Software Stack** describes the supported base software into which EJBCA has to be run.

Major functionality

EJBCA comes with two flavours: an Open Source version and an Enterprise Edition. Find below their major functionalities.

The Open Source version includes:

- Multiple CAs and levels of CAs: the possibility to build a complete infrastructure (or several) within one instance of EJBCA
- Support of most PKI protocols and algorithms considered safe today
- Support for Hardware Security Modules (HSMs)
- Individual enrollment or batch production of certificates
- Administrator registration and self-registration work-flows out of the box
- Browser enrollment through Firefox, IE, etc. with support for SmartCards
- Key recovery to store private keys for later recovery, for selected users and certificates
- OCSP and CRL services
- Support of special protocols like CMP, XKMS and SCEP
- Source code under LGPL license

The Enterprise Edition adds:

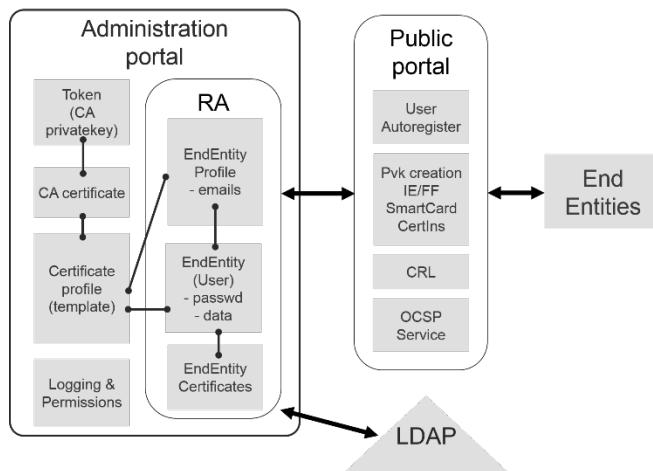
- Database integrity
- Secured audit logging
- Support for highly specialized deployments
- Professional support and know-how

Operation web portals

EJBCA is a software intended to build a full-compliant PKI infrastructure. In order to do so, it offers the following components:

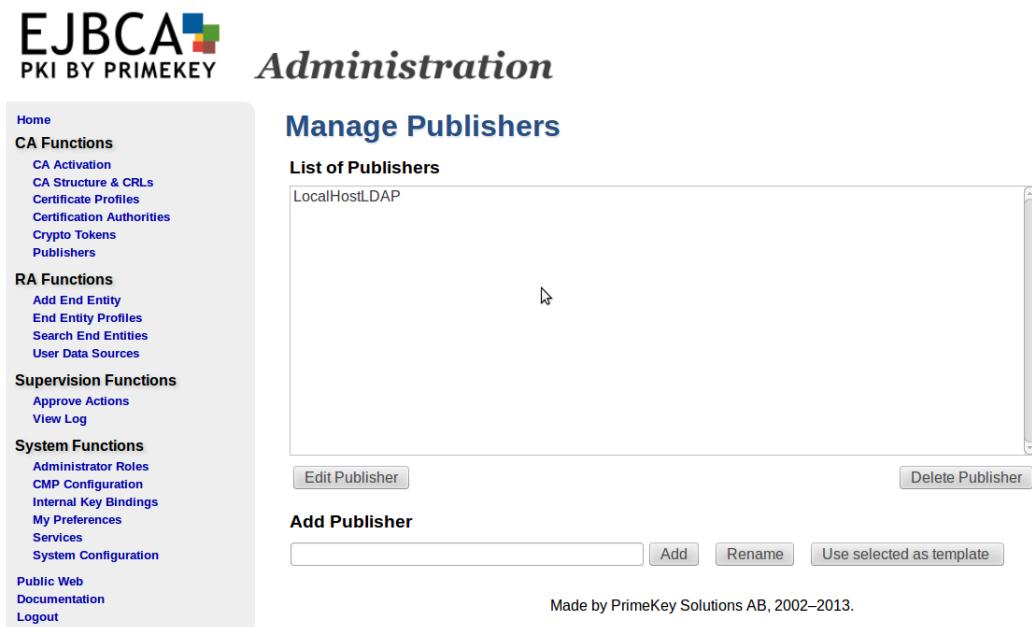
- A **Crypto Token** is a container of private key pairs. It can be stored in software (in the CA host) or in a Hardware Security Module like Thales nShield. More than one key can be stored in a Crypto Token because, aside from the CA signing key, it is useful to store another test key (to perform test signatures in order to check if the HSM module is alive).

- A **Certification Authority** contains a Crypto Token to perform signatures, the CA certificate, and the configuration of EJBCA embedded validation services (it specifies where the CRLs are published or where the OCSP service is located). It can handle multiple root Certification Authorities and their subordinated CAs.
- A **Certificate policy** specifies a certificate template containing all the fields of a certificate (CRL and OCSP issuing points, key sizes, algorithms, extensions, key usage, etc..) except for the subject of the certificate, and the CAs allowed to issue such certificate.
- **End Entities** are similar to end-users (be them individuals or servers) and have attributes such as user names, passwords, email addresses, etc. A CA will issue a certificate for an End Entity by attaching its identity or attributes to the issued certificate. End Entities can be supplied from an external source.
- **An End Entity profile** is an End Entity template that links the concept of CA and Certificate Policy with a user. It specifies the fields of each End Entity (such as address, email address, organization, name, phone number, and if the user password is auto generated) and how these fields are placed into the resulting Subject DN of the created certificate (the field can be fixed or modifiable). This profile can also include the need of informing an End Entity via email of a change in a certificate status.
- **A Registration Authority** manages End Entities and their profiles.
- **Publishers:** EJBCA can publish the issued certificates to a directory server like an LDAP or Active Directory in order to be visible for the organization.
- **Services:** EJBCA has scheduled internal services to send remainder emails to users if their certificate is about to expire, to renew CA keys, to renew CRLs automatically or to publish certificates.
- Various **Administration Roles** and complex security parameterisations can be managed.
- Various **Approvers** can be instructed to perform certain operations, thus enforcing the separation of duties.
- A **public portal** offers direct access to: revocation services (OCSP and CRL) and retrieval of CA certificates. It is also the main place where end users generate and retrieve certificates. This portal is equivalent to the RA portal in a PKI infrastructure.



Private administration portal

The private CA portal, usually only accessible via Administrative SSL client certificate in <https://server:8443/ejbca/adminweb>, contains the following options in its main menu:



The screenshot shows the EJBCA Administration interface. The left sidebar contains a navigation menu with the following items:

- Home
- CA Functions**
 - CA Activation
 - CA Structure & CRLs
 - Certificate Profiles
 - Certification Authorities
 - Crypto Tokens
 - Publishers
- RA Functions**
 - Add End Entity
 - End Entity Profiles
 - Search End Entities
 - User Data Sources
- Supervision Functions**
 - Approve Actions
 - View Log
- System Functions**
 - Administrator Roles
 - CMP Configuration
 - Internal Key Bindings
 - My Preferences
 - Services
 - System Configuration
- Public Web
- Documentation
- Logout

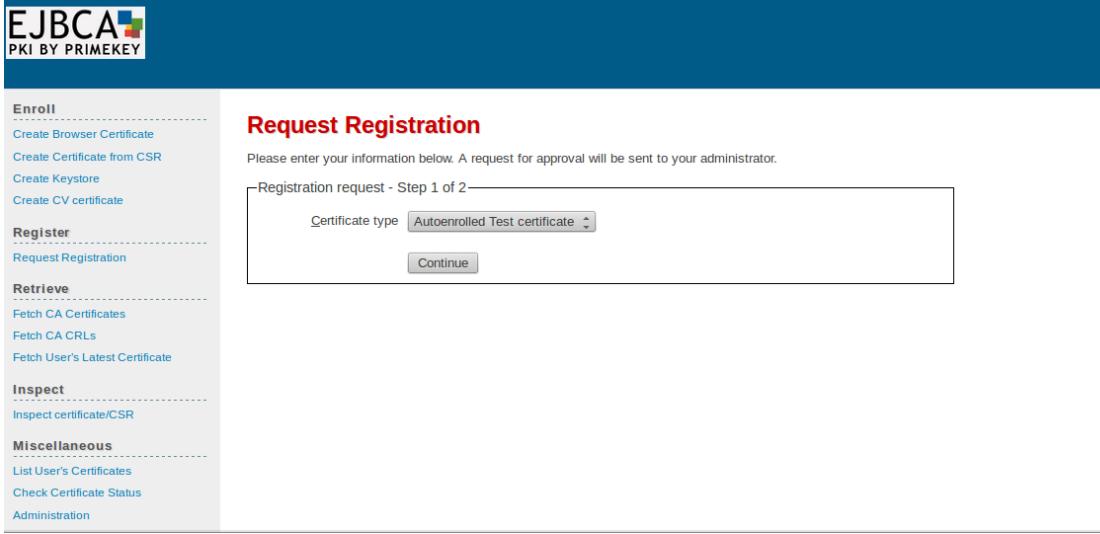
The main content area is titled "Manage Publishers". It displays a list of publishers with "LocalHostLDAP" listed. Below the list are two buttons: "Edit Publisher" and "Delete Publisher". At the bottom, there is a section for adding a new publisher with fields for "Name", "Add", "Rename", and "Use selected as template". A copyright notice at the bottom right states "Made by PrimeKey Solutions AB, 2002–2013."

- **[CA] CA activation** activates and deactivates CAs. This is useful to check if the private keys are active. It is also helpful when you are renewing the infrastructure.
- **[CA] CA Structure & CRLS** shows the hierarchical structure of the certification authorities managed by this instance of EJBCA (similar to the public portal).
- **[CA] Certificate Profiles** manages the templates for the CA issued certificates.
- **[CA] Certification Authorities** creates, revokes and deletes root and subordinate CAs.
- **[CA] Crypto Tokens** manages the private keys of the certification authorities.
- **[CA] Publishers** configures where the CA and the CA issued certificates are published.
- **[RA] Add End Entity** adds an end-user for which to issue certificates.
- **[RA] End Entity Profiles** manages End Entities and their data in order for EJBCA to issue certificates for them.
- **[RA] Search End Entities** searches for End Entities using various filters.
- **[RA] User Data Sources** configures plug-ins to use End Entities from an external source.
- **[Supervision] Approve Actions** configures actions that require more than one person to be done.
- **[Supervision] View Log** views log records.
- **[System] Administration Roles** assigns various permission types in the Administration console depending on some fields of the SSL logged certificate.
- **[System] CMP configuration** configures the CMP protocol. This protocol is used in large CA deployments.
- **[System] Internal Key Bindings** instructs EJBCA to use of private keys (Crypto Token) for purposes other than issuing certificates.
- **[System] My Preferences** configures user preferences.

- **[System] Services** configures scheduled background actions automatically performed by EJBCA.

Public administration portal

The public CA portal, usually accessible via <http://server:8080/ejbca> or <https://server:8442/ejbca> contains the following options in its main menu:



The screenshot shows the EJBCA Public administration portal. The top navigation bar has the EJBCA logo and the text "PKI BY PRIMEKEY". The main menu on the left includes sections for Enroll, Register, Retrieve, Inspect, and Miscellaneous, each with sub-options like "Create Browser Certificate", "Request Registration", etc. The central content area is titled "Request Registration" and displays a form for a "Registration request - Step 1 of 2". The form includes a dropdown for "Certificate type" set to "Autoenrolled Test certificate" and a "Continue" button. Below the form, a note says "Please enter your information below. A request for approval will be sent to your administrator."

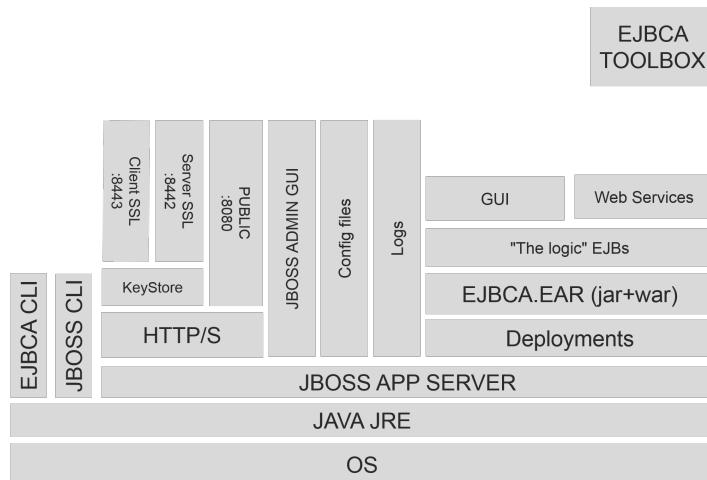
- **[Enroll] Create Browser Certificate** generates a certificate by entering an End Entity's user and password. The key pair can be generated via the browser ("User Generated" according to EJBCA) or in the server and downloaded as a PKCS#12 file ("P12" according to EJBCA). If IE is used, private keys can be created in smart cards using the "User Generated" option.
- **[Enroll] Create Certificate from CSR** generates a certificate by entering an End Entity's user and password and a Certificate Signing Request. This is generally used to generate certificates for SSL servers.
- **[Enroll] Create KeyStore** generates a Java KeyStore by entering an End Entity's user and password. The key pair is generated in the server.
- **[Enroll] Create CV Certificate** creates a Card Verifiable Certificate which is certificate designed to be processed by devices with limited computing power such as smart cards.
- **[Register] Request Registration** requests an RA to create a user for which to issue certificates.
- **[Retrieve] Fetch CA CRLs** downloads the CA CRLs.
- **[Retrieve] Fetch User's Latest Certificate** retrieves the last certificate issued for a user.
- **[Inspect] Certificate/CSR** examines the contents of a certificate or a Certificate Signing Request.
- **[Miscellaneous] List User's Certificates** retrieves any certificate (valid or not) issued for a certain user or End Entity. This can be useful when verifying long-term signatures.
- **[Miscellaneous] Check certificate status** checks if a certificate has been revoked.

System operation

EJBCA supports the following elements for system operation:

- Command line utility **bin/ejbca.sh**
- Embedded Web Services via Client Toolbox, an utility to call the Web Services from the command line
- Saves logs to a file although it can also be redirected to a log server to check errors
- Contains an integrated health check system

To better understand how EJBCA is built internally see this block diagram about its internal structure when deployed in a JBOSS Application Server:



- JBOSS CLI: command line utility to send locally commands to JBOSS, e.g. shutting down the server
- EJBCA CLI: command line utility to send locally commands to EJBCA, e.g. listing registered End Entities
- EJBCA TOOLBOX: utility to call EJBCA web services
- Public 8080: http server
- KeyStore: where the certificate and the private keys are located In JBOSS, usually in **jboss/standalone/configuration/keystore**
- Client SSL 8443: https server that requires a client certificate
- Server SSL 8443: https server that doesn't require a client certificate
- JBOSS ADMIN GUI: administration GUI for JBOSS, Very minimal. It can be accessed via <http://server:8080/>
- Log files: JBOSS log files are located in **jboss/standalone/logs**
- Config files: JBOSS configuration files are located in **jboss/standalone/configuration/standalone.xml**
- Deployments: the deployed applications in JBOSS are located in **jboss/standalone/deployments**

Integration with other components

EJBCA has connectors to various components.

- It uses services to:
 - Publish to an external LDAP or Active Directory server

- Import users from other sources
- Send e-mails
- Print
- Support Hardware Security Modules
- Create certificates in smart cards using the user's browser
- It offers services for:
 - OCSP (Online revocation check)
 - CRL (Downloadable revoked certificates blacklist)
 - CMP (PKI protocol for large deployments)
 - SCEP (Certificate Enrolment Protocol for devices)
 - XKMS (Protocol for key distribution)
 - RA Web Services
 - Administration Web Services

EJBCA has a plug-in architecture and the source code is available, so there are many options to connect to other systems.

Supported software stack

EJBCA supports the following stack:

- Java
 - Oracle Java 6/7
- Java Application JEE Server
 - JBoss 7.1.1.Final
 - JBoss EAP
 - JBoss 5.1.0.GA
 - JBoss EAP 5.1
- Databases
 - DB2
 - Oracle
 - MySQL
 - HyperSQL (local file)
- As EJBCA is built on JEE, it is transaction-safe and can be deployed in High Availability

Installing EJBCA and issuing self-signed certificates

Self-issued certificates are certificates that are signed by themselves. Within a PKI usually only CA Root certificates use this mechanism although it can also be used to issue SSL server certificates or other types of certificates out of the scope of a Certification Authority itself.

EJBCA is not designed for issuing standalone-self issued certificates, but this can be done easily. This section talks about how to use EJBCA to create self-signed certificates with private keys stored in software.

If you want to use an EJBCA to issue self-signed certificates and you have a production EJBCA PKI, it is recommended to use separated instances of EJBCA.

In this chapter we'll see how to:

- Install the EJBCA software
- Create the certificate profile SELF_ISSUED_SSL_SERVER: a template for issuing self-signed SSL Server certificates
- Create the CN=mysslserver.com self-signed certificate and download it with its private key as a PKCS#12 file

Installation

In order to install the EJBCA software we need to install Java and the Ant build tool in our server. In Debian hosts (like Ubuntu), this can be done with the following instructions:

```
sudo apt-get install openjdk-7-jdk
sudo apt-get install ant
```

Depending on the corporative system administration guidelines, this software can be located in different locations. In this case we use the `/opt` folder that is a common standard when manually installing software on UNIX machines. We need to download the JBoss Application Server 7.1.1 Final and the EJBCA Community Edition 6.0:

```
mkdir /opt
mkdir /opt/h2
unzip jboss-as-7.1.1.Final.zip
unzip ejbca_ce_6_0_3.zip
```

We need to set some configuration files:

Copy or rename `/opt/ejbca/conf/ejbca.properties.sample` to `/opt/ejbca/conf/ejbca.properties`
and set

`appserver.home=/opt/jboss-as-7.1.1.Final`

If your server hostname is not localhost copy or rename
`/opt/ejbca/conf/web.properties.sample` to `/opt/ejbca/conf/web.properties`
and set
`httpsserver.hostname=<hostname>`

Copy or rename `/opt/ejbca/conf/database.properties.sample` to `/opt/ejbca/conf/database.properties`

and set

`database.url=jdbc:h2:/opt/h2/ejbca;DB_CLOSE_DELAY=-1;MVCC=true`

Go to `/opt/jboss-as-7.1.1.Final/bin`, and run

`./standalone.sh`

Open a new terminal tab and go to `/opt/ejbca_ce_6_0_3`, then run

ant deploy

Sometimes, the first deployment is not successful, so it is necessary to repeat the **ant deploy** operation. When successful, run

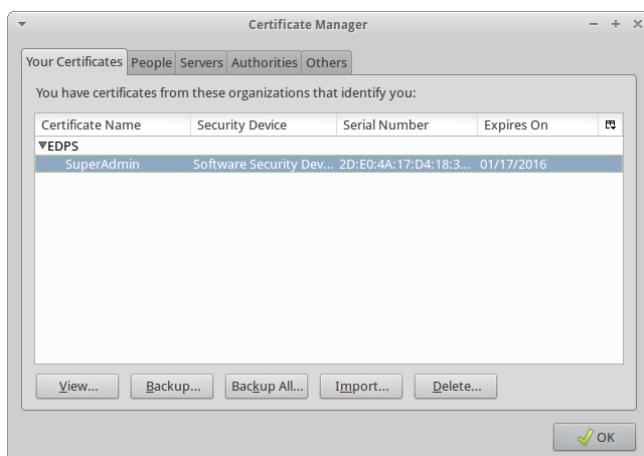
ant install

This will create a temporal administrative CA, accept all default values. When finished, press Control-C to stop the JBOSS Server, and start it again running `./standalone.sh`

Open the Firefox browser, go to Edit > Preferences > Advanced > View certificates>Import

Select the `/opt/ejbca_ce_6_0_3/p12/superadmin.p12` file. Its password is `ejbca`

This will import the SuperAdministrator certificate into the Firefox browser.



Now, you can enter the Administration portal through `https://localhost:8443/ejbca/adminweb` to

Creation of a certificate template

On the Administration portal, select **Certificate Profiles** from the menu. The next screen will appear:

Manage Certificate Profiles

List of Certificate Profiles

ENDUSER (FIXED)
OCSPSIGNER (FIXED)
ROOTCA (FIXED)
SERVER (FIXED)
SUBCA (FIXED)

Add Profile

Enter **SELF_ISSUED_SSL_SERVER** in the **Add Profile** textbox. Then, select the **SERVER (FIXED)** existing profile and press the **Use selected as template** button. This will create a certificate profile based on this template. This is really useful because we don't need to know the exact certificate key usages required to issue SSL server certificates. We cannot use the **SERVER (FIXED)** directly because it is not intended to issue self-signed certificates, so we must edit the created profile to adjust it. We can do it by selecting the **SELF_ISSUED_SSL_SERVER** and clicking the **Edit Certificate Profile** button.

Manage Certificate Profiles

List of Certificate Profiles

ENDUSER (FIXED)
OCSPSIGNER (FIXED)
ROOTCA (FIXED)
SELF_ISSUED_SSL_SERVER
SERVER (FIXED)
SUBCA (FIXED)

Change the **Type** to **Root CA**, and leave the other parameters as appear. Then click the **Save** button.

Edit Certificate Profile

Certificate Profile : SELF_ISSUED_SSL_SERVER

Certificate Profile Id		1480401626	Back to Certificate Profiles
Type		Root CA	Edit
Available bit lengths		0 bits 192 bits 239 bits 256 bits 384 bits	
Signature Algorithm		Inherit from issuing CA Edit	
Validity (*y *mo *d) or end date of the certificate [?]		730d <small>ISO 8601 date: YYYY-MM-DD HH:mm:ssZ730 2014-01-29 00:54:40 +00-00</small>	

Creation and exportation of a certificate

Go to the **Crypto Tokens** main menu option. The following screen will appear:

Manage Crypto Tokens [?]

Name	Type	Library	Reference Type	Reference	Active	Used	Action [?]
ManagementCA	Soft					Yes	Reactivate Delete

[Create new...](#)

Made by PrimeKey Solutions AB, 2002–2013.

Then click **Create new...**

New Crypto Token

Name:	<input type="text" value="myserverssl.com"/>
Type:	<input type="button" value="SOFT"/>
Authentication Code:	<input type="text" value="...."/>
Repeat Authentication Code:	<input type="text" value="...."/>
Auto-activation:	<input type="checkbox"/>
Allow export of private keys [?]:	<input checked="" type="checkbox"/>
Save	

Made by PrimeKey Solutions AB, 2002–2013.

Set the **Name** to **myserverssl.com**, the **Type** to **SOFT**, repeat the authentication code (this is the password protecting the private key), and check **Allow export of private keys** (it is not checked by default). Then, click the **Save** button.

Crypto Token myserverssl.com

CryptoToken created successfully.

[Back to Crypto Token overview](#) [Switch to edit mode](#)

Id:	57942174
Name:	myserverssl.com
Type:	SoftCryptoToken
Used:	<input type="checkbox"/>
Active:	<input checked="" type="checkbox"/>
Auto-activation:	<input type="checkbox"/>
Allow export of private keys [?]:	<input checked="" type="checkbox"/>

Crypto Token currently does not contain any key pairs.

RSA 2048 [Generate new key pair](#)

Made by PrimeKey Solutions AB, 2002–2013.

The container for private keys has been created. You should create now the private key itself. Select **RSA 2048** in the combo box and click **Generate new key pair**.

Crypto Token myserverssl.com

[Back to Crypto Token overview](#) [Switch to edit mode](#)

Id:	57942174
Name:	myserverssl.com
Type:	SoftCryptoToken
Used:	<input type="checkbox"/>
Active:	<input checked="" type="checkbox"/>
Auto-activation:	<input type="checkbox"/>
Allow export of private keys [?]:	<input checked="" type="checkbox"/>

	Alias	Key Algorithm	Key Specification	SubjectKeyID	Action
<input type="checkbox"/>	privatesignkeyalias	RSA	2048	063fbe0fc59c0ce008c4cd484e566db73773ed1d	Test Remove Download Public Key

[Remove selected](#)

[privatesignkeyalias](#) [RSA 2048](#) [Generate new key pair](#)

As you can see, you have a key pair created in a Crypto Token named **myserverssl.com**.

Now you should create the certificate. Go to the **Certification Authorities** menu and create a new Certification Authority called **myserverssl.com**:

Manage Certification Authorities

List of Certification Authorities

ManagementCA, (Active)

[Edit CA](#) [Delete CA](#) [Import CA keystore...](#) [Import CA certificate...](#)

[Create Authenticated Certificate Signing Request \[?\]](#)

Add CA

[Create...](#) [Rename](#)

Select it and click the **Create...** button. Select the mysslserver.com **Crypto Token**.

Create CA

CA Name : myserverssl.com

Back to Certificate Authorities	
Type of CA [?]	X509
Signing Algorithm	SHA1WithRSA
Crypto Token [?]	mysslserver.com
defaultKey:	privatesignkeyalias
certSignKey:	privatesignkeyalias
crlSignKey:	Use same as Certificate Signing Key.
keyEncryptKey:	- Use default key
hardTokenEncrypt:	- Use default key

Set the **Validity** to 2y, the **Subject DN** to the CN=mysslserver.com, **Signed By** to Self Signed and the **Certificate Profile** to SELF_ISSUED_SSL_SERVER.

Use Certificate Request History [?]	<input type="checkbox"/> Use
Use User Storage [?]	<input checked="" type="checkbox"/> Use
Use Certificate Storage [?]	<input checked="" type="checkbox"/> Use
CA certificate data	
Validity (*y *mo *d) or end date of the certificate [?]	2y ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2014-01-28 11:12:52-08:00'
Subject DN	CN=myserverssl.com
Signed By	Self Signed
Certificate Profile	SELF_ISSUED_SSL_SERVER

Press the Create button to create the Certificate:

- [Home](#)
- CA Functions**
 - [CA Activation](#)
 - [CA Structure & CRLs](#)
 - [Certificate Profiles](#)
 - [Certification Authorities](#)
 - [Crypto Tokens](#)
 - [Publishers](#)
- RA Functions**
 - [Add End Entity](#)
 - [End Entity Profiles](#)
 - [Search End Entities](#)
 - [User Data Sources](#)
- Supervision Functions**
 - [Approve Actions](#)
 - [View Log](#)
- System Functions**
 - [Administrator Roles](#)

Manage Certification Authorities

List of Certification Authorities

ManagementCA, (Active)
myserverssl.com, (Active)

[Edit CA](#) [Delete CA](#) [Import CA keystore...](#) [Import CA certificate...](#)

Select the certificate and click **Edit CA**. Go to the **Export CA** section, enter the protection password written in the creation of the Crypto Token, and select **Export CA keystore...** to download the PKCS#12 file. This file is password-protected (the same above mentioned password).

Export CA	
CA export requires the token authentication code	
****	Export CA keystore...

Basic Certification Authority

In the previous chapter we created SSL self-signed certificates, but this is not PKI. Now, we will create a complete Certification Authority that issues and revokes certificates, offers CRLs and OCSP services.

In this chapter, we will:

- Create a root Certification Authority with these parameters:
 - Name: CN=EDPSROOT, O=EDPS
 - Validity: 10 years
 - SHA256WithRSA signing algorithm
 - It issues CRLs
 - It has an OCSP server
- Create 2 types of certificates and End Entity profiles
 - SSL Client certificate
 - We create the Certificate Profile EDPS_SSLCLIENT_CERTPROFILE that will specify the certificate template for an SSL Client certificate.
 - We create the End Entity Profile EDPS_SSLCLIENT_EEP PROFILE that forces that:
 - The subject of the certificate is CN=<id of user>, O=EDPS.
 - The private key can be generated by the server (this is, the user downloads a PKCS#12), or is supplied by the user.
 - SSL Server certificate
 - We create the Certificate Profile EDPS_SSLSERVER_CERTPROFILE that will specify the certificate template for an SSL Server certificate.
 - We create the End Entity Profile EDPS_SSLSERVER_EEP PROFILE that forces that:
 - The subject of the certificate is CN=<id of host>, O=EDPS.
 - The private key can be generated by the server (this is, the user downloads a JKS java keystore), or supplied by the server.
- Create three End Entities and generate a certificate for each one of them:
 - **CN=john, O=EDPSCA** SSL Client certificate. Here the user will download the certificate from the CA in a PKCS#12 file.
 - **CN=peter, O=EDPSCA** SSL Client certificate. The user will generate the private key in the browser (this is, in the client-side), and the CA will generate the certificate with the supplied public key.
 - **CN=myserverssl.com, O=EDPS** SSL Server certificate. The certificate will be generated from a CSR that has been generated in the command line using the openssl tool.
- Revoke the **CN=john** certificate and regenerate the CRL

Creation of the EDPSROOT Certification Authority

To create the Root Certification Authority go to **Certification Authorities** and create the **EDPSROOT** Certification Authority by entering the name in the **Add CA** textbox and pressing the **Create...** button.

Add CA

EDPSROOT	<input type="button" value="Create..."/>	<input type="button" value="Rename"/>
----------	--	---------------------------------------

Select **SHA256WithRSA** as the **Signing Algorithm**, and in **Crypto Token** choose the default option – **Create a new soft Crypto Token with recommended key pairs**.

Create CA

CA Name : EDPSROOT

Type of CA [?]	X509	<input type="button" value="Back to Certificate Authorities"/>
Signing Algorithm	SHA256WithRSA	
Crypto Token [?]	- Create a new soft Crypto Token with recommended key pairs	

We need to supply the **Subject DN** of this CA: CN=EDPSROOT,O=EDPS , the **Validity** to 10y and the **Certificate Profile** to ROOTCA.

CA certificate data	
Validity ('y *mo *d) or end date of the certificate [?]	10y ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2014-01-29 02:39:14-08:00'
Subject DN	CN=EDPSROOT, O=EDPS
Signed By	Self Signed
Certificate Profile	ROOTCA

We also need to configure where the CRLs will be published: press each one of the **Generate** buttons in **Default CRL Dist. Point**, **Default CRL Issuer** and **CA Defined FreshestCRL extension**. By default, EJBCA chooses the localhost server, change it for the real server hostname, in this case **edpsca**.

CRL Specific Data	
Authority Key ID	<input checked="" type="checkbox"/> Use <input type="checkbox"/> Critical
CRL Number	<input checked="" type="checkbox"/> Use <input type="checkbox"/> Critical
Issuing Distribution Point on CRLs	<input type="checkbox"/> Use <input checked="" type="checkbox"/> Critical
Default CRL Dist. Point [?]	<input type="text" value="http://edpsca:8080/ejbca/publicweb/webdist/certdist?cmd="/> <input type="button" value="Generate"/> (used as default value in certificate profiles using this CA)
Default CRL Issuer [?]	<input type="text" value="CN=EDPSROOT,O=EDPS"/> <input type="button" value="Generate"/>
CA Defined FreshestCRL extension [?]	<input type="text" value="http://edpsca:8080/ejbca/publicweb/webdist/certdist?cmd="/> <input type="button" value="Generate"/> (used as default value in certificate profiles using this CA)

In the **Services** section, press **Generate Button** of the **Default OCSP Service Locator**. Replace the localhost parameter by the name of the host.

Services	
Default OCSP Service Locator	<input type="text" value="http://edpsca:8080/ejbca/publicweb/status/ocsp"/> [?]
	<input type="button" value="Generate"/>
(used as default value in certificate profiles using this CA)	

The EDPSROOT CA has been created. Go to the **Crypto Tokens** main menu to examine the created keys. There are two keys, the first one is named **signKey** or **defaultKey**, and the other one is **testKey**. In this case, the test key is not useful because it is in software; although it will be useful in HSM environments.

Crypto Token EDPSROOT

Back to Crypto Token overview [Switch to edit mode](#)

Id:	-1417010799			
Name:	EDPSROOT			
Type:	SoftCryptoToken			
Used:	<input checked="" type="checkbox"/>			
Active:	<input checked="" type="checkbox"/>			
Auto-activation:	<input checked="" type="checkbox"/>			
Allow export of private keys [?]:	<input type="checkbox"/>			
<hr/>				
Alias	Key Algorithm	Key Specification	SubjectKeyID	Action
<input type="checkbox"/> defaultKey	RSA	2048	d33d3f9fc2be1e8a8193a6203c103d8766dec9ee	<input type="button" value="Test"/> <input type="button" value="Remove"/> <input type="button" value="Download Public Key"/>
<input type="checkbox"/> signKey	RSA	2048	d3a1dc5c121296f2f6bc3d486be0035ebc86338b	<input type="button" value="Test"/> <input type="button" value="Remove"/> <input type="button" value="Download Public Key"/>
<input type="checkbox"/> testKey	RSA	1024	9e162fdeea180d80210e6feba6c9d5574f57ed9d	<input type="button" value="Test"/> <input type="button" value="Remove"/> <input type="button" value="Download Public Key"/>
<hr/>		<input type="button" value="Remove selected"/>		

If we go to **CA Structure & CRLs** we can see the created EDPSROOT CA and the last CRL created. Later on, we will use the **Create CRL** button in this screen to renew the CRL.

CA Structure & CRLs

Basic Functions for CA : EDPSROOT [View Certificate](#) [View Information](#)
 Root CA : CN=EDPSROOT,O=EDPS

[Download binary](#) [to IE](#) [Download to Firefox](#) [Download PEM file](#) [Download JKS file](#)

Latest CRL: Created 2014-01-20 12:15:57+00:00, Expires 2014-01-21 12:15:57+00:00, number 1 [Get CRL](#)
 Delta CRLs are not enabled.

Create a new updated CRL : [Create CRL](#)

Basic Functions for CA : ManagementCA [View Certificate](#) [View Information](#)
 Root CA : CN=ManagementCA,O=EDPS

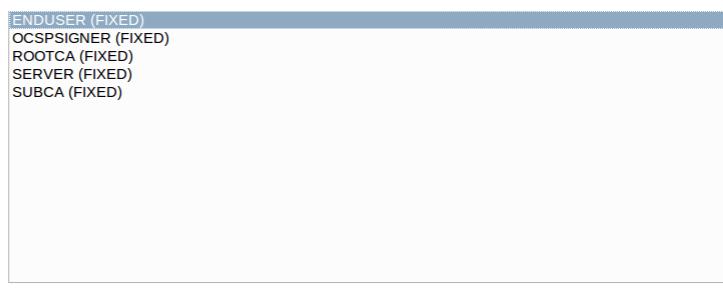
[Download binary](#) [to IE](#) [Download to Firefox](#) [Download PEM file](#) [Download JKS file](#)

Latest CRL: Created 2014-01-20 11:52:47+00:00, Expires 2014-01-21 11:52:47+00:00, number 1 [Get CRL](#)
 Delta CRLs are not enabled.

Creation of the certificate and End Entity profiles

Create the SSL client profile

Go to the **Certificate Profiles** main menu option, and create the **EDPS_SSLCLIENT_CERTPROFILE** using the **ENDUSER (FIXED)** profile with the **Use selected as template** button.



Select **EDPS_SSLCLIENT_CERTPROFILE** and press **Edit Certificate Profile**. Select **1024** and **2048** as Available bit lengths.

Edit Certificate Profile

Certificate Profile : EDPS_SSLCLIENT_CERTPROFILE

		Back to Certificate Profiles
Certificate Profile Id	1620831371	
Type	End Entity	
Available bit lengths	<input type="button" value="1024 bits"/> <input type="button" value="1536 bits"/> <input type="button" value="2048 bits"/> <input type="button" value="4096 bits"/> <input type="button" value="8192 bits"/>	
Signature Algorithm	Inherit from issuing CA	
Validity (*y *mo *d) or end date of the certificate [?]	730d ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ] '2014-01-20 12:23:42+00:00'	

In the **CRL Distribution Points** section, select the **Use CA defined CRL Dist. Point**. Check **Authority Information Access** and **Use CA defined OCSP locator**.

CRL Distribution Points [?]	<input checked="" type="checkbox"/> Use <input type="checkbox"/> Critical
Use CA defined CRL Dist. Point	<input checked="" type="checkbox"/> Use
CRL Distribution Point URI	<input type="text"/>
CRL Issuer [?]	<input type="text"/>
FreshestCRL extension [?]	<input type="checkbox"/> Use
Use CA Defined FreshestCRL extension	<input type="checkbox"/> Use
FreshestCRL extension URI	<input type="text"/>
Certificate Policies	<input type="checkbox"/> Use <input checked="" type="checkbox"/> Critical
Add	<input type="button"/>
Certificate Policy Id	<input type="text"/>
User Notice Text	<input type="text"/>
CPS URI	<input type="text"/>
Authority Information Access	<input checked="" type="checkbox"/> Use
Use CA defined OCSP locator	<input checked="" type="checkbox"/> Use
OCSP Service Locator URI [?]	<input type="text"/>

Finally, select that this certificate profile is only issuable from the **EDPSROOT** Certification Authority in **Available CAs**.

Available CAs	<input type="checkbox"/> Any CA <input checked="" type="checkbox"/> EDPSROOT <input type="checkbox"/> ManagementCA
---------------	--

SSL Server certificate profile

Create the **EDPS_SSLSERVER_CERTPROFILE** Profile following the same steps but using **SERVER (FIXED)** as template instead of **CLIENT (FIXED)**.

SSL Client entity profile

Go to **End Entity Profiles** and enter **EDPS_SSLCLIENT_EEPROFILE** in the **Add Profile** textbox.

Manage End Entity Profiles

List of End Entity Profiles

EMPTY

Edit End Entity Profile **Delete End Entity Profile**

Add Profile

EDPS_SSLCLIENT_EEPROFILE **Add** **Rename** **Use selected as template**

Click the **Add** button.

We need that all SSL client certificates are issued with the O=EDPS field in the Subject DN. To do so, add a field by selecting **Organization** in **Subject DN Attributes**, and pressing **Add**. Then, write **EDPS** in the **O, Organization** textbox, check **Required**, and uncheck **Modifiable**. Add the email field, also.

Select for Removal	Subject DN Attributes	
<input type="checkbox"/>	CN, Common name	<input type="text"/> emailAddress, E-mail address in DN Add
<input type="checkbox"/>	emailAddress, E-mail address in DN	<input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Modifiable <input type="checkbox"/> Required See also configuration of E-mail field.
<input type="checkbox"/>	O, Organization	<input type="text"/> EDPS <input checked="" type="checkbox"/> Required <input type="checkbox"/> Modifiable
Remove		

Finally, select that the default and available profile in **Main certificate data** is the **EDPS_SSLCLIENT_CERTPROFILE**, that the default and available CA is **EDPSROOT** and that the **Available Tokens** are **User Generated** and **P12 file**.

	Main certificate data
	Default Certificate Profile: EDPS_SSLCLIENT_CERTPROFILE Available Certificate Profiles: EDPS_SSLCLIENT_CERTPROFILE , EDPS_SSLSERVER_CERTPROFILE , ENDUSER , OCSPSIGNER , SERVER , SUBCA
	Default CA: EDPSROOT Available CAs: Any CA , EDPSROOT , ManagementCA
	Default Token: User Generated Available Tokens: User Generated , P12 file , JKS file , PEM file

Finally, press the **Save** button.

SSL Server entity profile

Create the **EDPS_SSLSERVER_EEPROFILE** profile with the same parameters as we created **EDPS_SSLCLIENT_EEPROFILE**, but unchecking **End Entity E-mail Use** and using **EDPS_SSLSERVER_CERTPROFILE** as the default and available certificate profile.

Edit End Entity Profile

End Entity Profile : **EDPS_SSLSERVER_EEPROFILE**

		Back to End Entity Profiles
	End Entity Profile Id	1962908462
	Username	<input type="text"/>
	Password (or Enrollment Code) [?]	<input type="password"/> <input checked="" type="checkbox"/> Required <input type="checkbox"/> Autogenerated Digits only of length <input type="text" value="8"/>
	Minimum password strength (bits) [?]	<input type="text" value="0"/>
	Maximum number of failed login attempts [?]	<input type="checkbox"/> Use : Default = <input type="radio"/> Unlimited <input checked="" type="checkbox"/> Modifiable
	Batch generation (clear text pwd storage)	<input type="checkbox"/> Use : Default = <input type="radio"/> Required <input type="checkbox"/> Use (Use only the domain part of the address, without the '@' char)
	End Entity E-mail	<input type="text"/> <input type="checkbox"/> Required <input type="checkbox"/> Modifiable

Issuing the certificates

Creating end-users

We need to create users, there's no possibility in EJBCA to create End Entity certificates without the "user" concept (in normal deployments, a special configuration allows it).

We will create three users:

- user **john** with **EDPS_SSLCLIENT_EEPROFILE** in **User Generated Token**
- user **peter** with **EDPS_SSLCLIENT_EEPROFILE** in **P12 file Token**
- user **myserversssl** with **EDPS_SSLSERVER_EEPROFILE** and **User Generated Token**

Add End Entity

End Entity Profile	EDPS_SSLCLIENT_EEPROFILE	<input checked="" type="checkbox"/> Required
Username	john	<input checked="" type="checkbox"/>
Password (or Enrollment Code)	<input checked="" type="checkbox"/>
Confirm Password	
E-mail address	john @ server.com	<input type="checkbox"/>
Subject DN Attributes		
CN, Common name	john	<input checked="" type="checkbox"/>
emailAddress, E-mail address in DN	<input checked="" type="checkbox"/> Use data from E-mail address field	<input type="checkbox"/>
O, Organization	EDPS	<input checked="" type="checkbox"/>
Main certificate data		
Certificate Profile	EDPS_SSLCLIENT_CERTPROFILE	<input checked="" type="checkbox"/>
CA	EDPSROOT	<input checked="" type="checkbox"/>
Token	User Generated	<input checked="" type="checkbox"/>
		Add Reset

End Entity Profile	EDPS_SSLCLIENT_EEPROFILE	Required
Username	peter	<input checked="" type="checkbox"/>
Password (or Enrollment Code)	***	<input checked="" type="checkbox"/>
Confirm Password	***	<input checked="" type="checkbox"/>
E-mail address	peter @ server.com	<input type="checkbox"/>
Subject DN Attributes		
CN, Common name	peter	<input checked="" type="checkbox"/>
emailAddress, E-mail address in DN	<input checked="" type="checkbox"/> Use data from E-mail address field	<input type="checkbox"/>
O, Organization	EDPS	<input checked="" type="checkbox"/>
Main certificate data		
Certificate Profile	EDPS_SSLCLIENT_CERTPROFILE	<input checked="" type="checkbox"/>
CA	EDPSROOT	<input checked="" type="checkbox"/>
Token	P12 file	<input checked="" type="checkbox"/>
	Add	Reset

Add End Entity

End Entity Profile	EDPS_SSLSERVER_EEPROFILE	Required
Username	myserverssl	<input checked="" type="checkbox"/>
Password (or Enrollment Code)	****	<input checked="" type="checkbox"/>
Confirm Password	****	<input checked="" type="checkbox"/>
Subject DN Attributes		
CN, Common name	mysslserver.com	<input checked="" type="checkbox"/>
Main certificate data		
Certificate Profile	EDPS_SSLSERVER_CERTPROFILE	<input checked="" type="checkbox"/>
CA	EDPSROOT	<input checked="" type="checkbox"/>
Token	User Generated	<input checked="" type="checkbox"/>
	Add	Reset

We can see the created users in the **Search End Entities** menu (in the **Filter** use **New in Search end entities with status**).

Select	Username	CA	CN	OU	O (organization)	Status
<input type="checkbox"/>	john	EDPSROOT	john		EDPS	New
<input type="checkbox"/>	myserverssl	EDPSROOT	mysslserver.com			New
<input type="checkbox"/>	peter	EDPSROOT	peter		EDPS	New

Select All Unselect All Invert Selection

Revocation reason :

Generating a client SSL certificate for john

Go the public web portal and click the **Public Web** option in the menu.

Then, in the public portal click the **Create Browser Certificate** menu, enter **john** and its password, and click the **OK** button.

Certificate Enrollment

Welcome to Certificate Enrollment.

Please enter your username and enrollment code. Then click OK to generate your token.

Authentication

Username	john
Enrollment code	****
<input type="button" value="OK"/>	

A key pair will be generated in the browser, and EJBCA will issue and install a certificate for this key pair. All these operations are done automatically with JavaScript (or VBScript in windows systems) in the browser. After the process, EJBCA shows you that the certificate has been created and installed in the browser.

Certificate Created

Subject DN: E=john@server.com,CN=john,O=EDPS

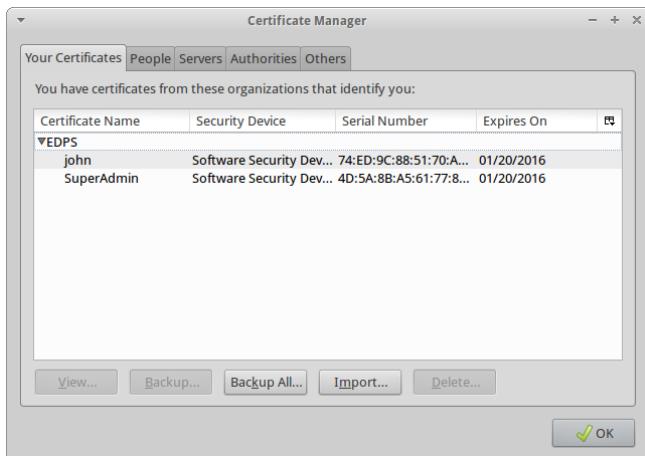
Issuer DN: CN=EDPSROOT,O=EDPS

Serial Number: 74E09C885170A6D7

If your certificate is not installed automatically, please click here to install it:

[Install certificate](#)

Open the Firefox browser, go to Edit > Preferences > Advanced > View certificates, and the created certificate will be shown.



Generating a client SSL certificate for peter

In the public portal click the **Create Browser Certificate** menu, enter **peter** and its password, and click the **OK** button.

EJBCA Token Certificate Enrollment

Welcome to keystore enrollment.

If you want to, you can manually install the CA certificate(s) in your browser, otherwise this will be done automatically when your certificate is retrieved.

Install CA certificates:

[Certificate chain](#)

Please choose a key length, then click OK to fetch your certificate.

Options

Leave values as default if unsure.

Key length:

You will be able to choose the **Key length** of the certificate in a menu. Then, press the **Enroll** button. EJBCA will create a PKCS#12 file downloadable with the browser.

Generating the SSL server certificate

SSL server certificates are usually created via CSR. CSR is a file format that contains the public key and the subject of the certificate. It is generated in the client side. The CSR is given to EJBCA and EJBCA issues a certificate with the supplied public key. Usually CAs also pick the subject from the CSR file, but EJBCA doesn't, that's why the issuer of the created certificate must be well pre-defined in EJBCA.

In this example, we use the **openssl** tool that is available in almost all UNIX systems. First, create the key pair with the **genrsa** parameter. Afterwards, use the **req** parameter to generate a CSR.

```
user@edpsca:~/mysslserver$ openssl genrsa -out mysslserver.key 2048
Generating RSA private key, 2048 bit long modulus
....+++
.....+e is 65537 (0x10001)
user@edpsca:~/mysslserver$ openssl req -new -key mysslserver.key -out server.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Barcelona
Locality Name (eg, city) []:Barcelona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:EDPS
Organizational Unit Name (eg, section) []:PKI
Common Name (e.g. server FQDN or YOUR name) []:mysslserver.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
user@edpsca:~/mysslserver$
```

If we dump the **server.csr** file, we can see that it is encoded in a base-64 format. Copy this text to the clipboard.

```
user@edpsca:~/mysslserver$ cat server.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICsTCACZkCAQAwDELMAKGA1UEBhMCRVmxEjAQBgNVBAgMCUJhcmN1bG9uYTES
MBAGA1UEBwwJQmFyY2Vs25hMQwCwYDVQQKDARFRFBTMQwxCgYDVQLDANQS0kx
GDAWBgNVBAMMD215c3Nsc2VydmlvLmNvbTCCAS1wDQYJKoZIhvNaQEBBQADggEP
ADCCAQoCggEBAM7Zc9x7fU15rpnMrpm2WBKaxXQxtAT4WDaKvNoGjw0qdgQM1omt
HesCpPt1N7upiVdb1GVsaS0mysJyQJ5pf7ym0Vgiok1TqmagY0vUUkmulv9EJfQ
iE0bie1as+9rCl2rq35zK25c+YaghH/gp9ITKOTlG1oBBSBoV700.0.JPxSF4rV
crlD79zcPdykITSRaBL1icd2l66RoIk/qETY6XIPCq5x+C1-S69Y5CV8H5t1tfV
JDZ1vh5T6j20jXeh17r1sl+3fzoqURp4oH+Y-QfrnwBMDFivhyhCUAnLPu8BjN
oZku1cke2zUKQ-/cPlommUcv0rhwoCNHBCAwEAaaAMAAQGCsgGS1b3DQEBBQUA
A41BAQBQCjRw5Efo0qBytgdW6tFb0XXHYK1cQ1qwtz5Gym881idektxnCavExZh
ag37Yihm2zeqn+0DkuRUNudMBFAAvyB5COpwxFx8.JHGRyU1NL1kp086xTTQ24A
Pj8G+8QXhyupt2weEtvdsvzLm4ycTzfkmKY08d+iwTKwZXue4vw+rf1Dof8/id
b7aqR6ghOptANFd7cz/iuh0/Rmc93MwxdhDUHqeKH2m3gt1M0A9M78DBnU
htwRh2Dv1Pcw2VTYuYnAs0j4QAHwDU+Ngnzrn7eZA8EIkvXRExU0oM9tVicZTdY
qFFXVDaUdDkpm1js/DYMa/MgorkN
-----END CERTIFICATE REQUEST-----
user@edpsca:~/mysslserver$
```

Go to the public portal, and select **Create Certificate from CSR**, enter the **myserverssl** user and password. Then, paste the server.csr text into the text area, select **PEM – certificate only**, and click the **OK** button.

Certificate enrollment from a CSR

Please give your username and enrollment code, select a PEM- or DER-formatted certification request file (CSR) for upload, or paste a PEM-formatted request into the field below and click OK to fetch your certificate.

A PEM-formatted request is a BASE64 encoded certificate request starting with
-----BEGIN CERTIFICATE REQUEST-----
and ending with
-----END CERTIFICATE REQUEST-----

Enroll

Username	<input type="text" value="myserverssl"/>
Enrollment code	<input type="text" value="*****"/>
Request file	<input type="button" value="Browse..."/> No file selected.
or pasted request	
<pre>-----BEGIN CERTIFICATE REQUEST----- MIICsTCACZkCAQAwDELMAKGA1UEBhMCRVmxEjAQBgNVBAgMCUJhcmN1bG9uYTES MBAGA1UEBwwJQmFyY2Vs25hMQwCwYDVQQKDARFRFBTMQwxCgYDVQLDANQS0kx GDAWBgNVBAMMD215c3Nsc2VydmlvLmNvbTCCAS1wDQYJKoZIhvNaQEBBQADggEP ADCCAQoCggEBAM7Zc9x7fU15rpnMrpm2WBKaxXQxtAT4WDaKvNoGjw0qdgQM1omt HesCpPt1N7upiVdb1GVsaS0mysJyQJ5pf7ym0Vgiok1TqmagY0vUUkmulv9EJfQ</pre>	
Result type	<input type="button" value="PEM - certificate only"/>
<input type="button" value="OK"/>	

The **myserver.pem** file is downloaded. This PEM file is an X509v3 file encoded in base64.

Revoking john's certificate and generating a new CRL

To revoke a certificate, search the End Entity that contains the certificate you want to revoke in the **Search End Entities** menu. In this case we use **john**, so we check the **Select** checkbox, choose the **Revocation reason** (in this case **Cessation of operation**), and click the **Revoke Selected** button.

Search End Entities

[Advanced Mode](#)

Search end entity with username	<input type="text" value="john"/>	<input type="button" value="Search"/>																																						
Search end entity with Certificate SN (hex)	<input type="text"/>	<input type="button" value="Search"/>																																						
Search end entities with status	<input type="button" value="..."/>	<input type="button" value="Search"/>																																						
Search end entities with certificates expiring within	<input type="text"/> Days	<input type="button" value="Search"/>																																						
<input type="button" value="Reload"/>																																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Select</th> <th style="width: 20%;">Username</th> <th style="width: 10%;">CA</th> <th style="width: 10%;">CN</th> <th style="width: 10%;">OU</th> <th style="width: 10%;">O (organization)</th> <th style="width: 10%;">Status</th> <th style="width: 10%;"></th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/></td> <td>john</td> <td>EDPSROOT</td> <td>john</td> <td></td> <td>EDPS</td> <td>Generated</td> <td style="text-align: right;"> View End Entity Edit End Entity View Certificates View History </td> </tr> <tr> <td colspan="7" style="text-align: center; padding-top: 5px;"> <input type="button" value="Select All"/> <input type="button" value="Unselect All"/> <input type="button" value="Invert Selection"/> </td> </tr> <tr> <td colspan="3" style="text-align: left; padding-top: 10px;"> <input type="button" value="Revoke Selected"/> <input type="button" value="Revoke And Delete"/> </td> <td colspan="4" style="text-align: right;"> <input type="button" value="Delete Selected"/> </td> </tr> <tr> <td colspan="8" style="text-align: center; padding-top: 5px;"> Revocation reason : <input type="button" value="Cessation of operation"/> </td> </tr> </tbody> </table>			Select	Username	CA	CN	OU	O (organization)	Status		<input checked="" type="checkbox"/>	john	EDPSROOT	john		EDPS	Generated	View End Entity Edit End Entity View Certificates View History	<input type="button" value="Select All"/> <input type="button" value="Unselect All"/> <input type="button" value="Invert Selection"/>							<input type="button" value="Revoke Selected"/> <input type="button" value="Revoke And Delete"/>			<input type="button" value="Delete Selected"/>				Revocation reason : <input type="button" value="Cessation of operation"/>							
Select	Username	CA	CN	OU	O (organization)	Status																																		
<input checked="" type="checkbox"/>	john	EDPSROOT	john		EDPS	Generated	View End Entity Edit End Entity View Certificates View History																																	
<input type="button" value="Select All"/> <input type="button" value="Unselect All"/> <input type="button" value="Invert Selection"/>																																								
<input type="button" value="Revoke Selected"/> <input type="button" value="Revoke And Delete"/>			<input type="button" value="Delete Selected"/>																																					
Revocation reason : <input type="button" value="Cessation of operation"/>																																								

After that, the **Status** of the certificate changes to **Revoked**.

Select	Username	CA	CN	OU	O (organization)	Status	
<input type="checkbox"/>	john	EDPSROOT	john		EDPS	Revoked	View End Entity Edit End Entity View Certificates View History

If you click the **View Certificate** link, more information about the revocation is shown.

View Certificate

Username	john
Certificate number	1 of 1
Certificate Type/Version	X.509 v.3
Certificate Serial Number	74ED9C88517046D7
Issuer DN	CN=EDPSROOT,O=EDPS
Valid from	2014-01-20 15:05:18+00:00
Valid to	2016-01-20 15:05:18+00:00
Subject DN	E=john@server.com,CN=john,O=EDPS
Subject Alternative Name	None
Subject Directory Attributes	None
Public key	RSA (2048 bits): C15DB69E2EE5813B96A554B0A60A97F69CF8C00D8DDE25F...
Basic constraints	End Entity
Key usage	Digital Signature, Non-repudiation, Key encipherment
Extended key usage	Client Authentication, Email Protection
Qualified Certificates Statements	No
Signature Algorithm	SHA256WITHRSA
Fingerprint SHA-1	A2BE3683351C476CC262F16844E46CA65F40FD5B
Fingerprint MD5	884B77BA9F2A6EEEC469E9EBD7ED9B54
Revoked	Yes
	Revocation date : 2014-01-20 15:53:56+00:00
	Revocation reasons : Cessation of operation
<input type="button" value="Republish"/> <input type="button" value="Close"/>	
<input type="button" value="Download binary/IE"/> <input type="button" value="Download to Firefox"/> <input type="button" value="Download PEM file"/> <input type="button" value="Download JKS file"/>	

To renew the CRL, go to the **CA Structure & CRLs** main menu option and click the **Create CRL** button in the **EDPSROOT** CA.

CA Structure & CRLs

Basic Functions for CA : EDPSROOT [View Certificate](#) [View Information](#)

Root CA : CN=EDPSROOT,O=EDPS

[Download binary/IE](#) [Download to Firefox](#) [Download PEM file](#) [Download JKS file](#)

Latest CRL: Created 2014-01-20 15:56:35+00:00, Expires 2014-01-21 15:56:35+00:00, number 2 [Get CRL](#)
 Delta CRLs are not enabled.

Create a new updated CRL :

Advanced Certification Authority

A Certification Authority may have more specialised features than the CA created in the previous chapter. For example:

- Two subordinated CAs exist: one for issuing administrator certificates and the other one for issuing end user certificates.
- Certain CA administrators ARE NOT super administrators of the whole EJBCA.
- Certificates are published in a corporative directory.
- Users can enroll themselves instead of being previously created. In this case, emails are sent to users.
- Dual-control is required to issue certificates.
- Certificates have a policy to identify its own usage.
- CA data can be retrieved from another server.

In this chapter these functionalities will be implemented:

- Two subordinated CAs will be created:
 - EDPSADMIN will be created to issue only administration certificates
 - SuperAdministrator certificates
 - CA administrator certificates
 - EDPSNQC will be created to issue non-qualified certificates
 - SSL client certificates with policy 1.3.130.2.1: self-enrolled users with subject CN=<user>, OU=NQC, O=EDPS.
 - SSL client certificates with policy 1.3.130.2.2: they need the approval of two RA administrators. Their subject will be CN=<user>, FirstName=<initials>, SurName=<surname>, UID=<personalID>, OU=NQC, O=EDPS.
- Two 1.3.130.2.1 certificates will be created and permissions will be given to approve 1.3.120.2.2 certificates.
- Certificates will be published to an LDAP server.

And with the following implementation:

- Two 1.3.130.2.1 certificates will be created: one for CN=ejbcauser1 and the other one for CN=ejbcauser2, both self-enrolled in the public portal with password sent by email.
- Both certificates will be configured as valid RA Administrators.
- A 1.3.130.2.2 user with CN=ejbcauser3 will be issued with the dual approbation of CN=ejbcauser1 and CN=ejbcauser2.
- All certificates will be published into an LDAP server.
- Finally we will access the EJBCA Web Services from a remote machine.

Please, note that this configuration is only for training purposes and cannot be used in the real world due to lack of security.

Create the administration and NQC subordinated CAs

Go to **Certification Authorities**, and create the **EDPSADMIN** certification authority. Then select SHA256WithRSA as the **Signing Algorithm**.

Create CA

CA Name : EDPSADMIN

		Back to Certificate Authorities
Type of CA [?]	X509	
Signing Algorithm	SHA256WithRSA	
Crypto Token [?]	- Create a new soft Crypto Token with recommended key pairs	
Key sequence format [?]	numeric [0-9]	
Key sequence [?]	00000	
Description		

Set the **Validity** to 5 years and the **Subject DN** to CN=EDPSADMIN, O=EDPS. Choose EDPSROOT in **Signed By** and SUBCA in **Certificate Profile**.

Validity (*y *mo *d) or end date of the certificate [?]	5y
ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2014-01-28 10:27:00-08:00'	
Subject DN	CN=EDPSADMIN, O=EDPS
Signed By	EDPSROOT
Certificate Profile	SUBCA

We need to configure where the CRLs will be published. Press the **Generate** button in **Default CRL Dist.Point**, **Default CRL Issuer** and **CA Defined FreshestCRL extension**. By default, they will be placed in the localhost server, change it by the server hostname, in this case **edpsca**.

Default CRL Dist. Point [?]	http://edpsca:8080/ejbca/publicweb/webdist/certdist?
<input type="button" value="Generate"/>	
(Used as default value in certificate profiles using this CA)	
Default CRL Issuer [?]	CN=EDPSADMIN, O=EDPS
<input type="button" value="Generate"/>	
CA Defined FreshestCRL extension [?]	http://edpsca:8080/ejbca/publicweb/webdist/certdist?
<input type="button" value="Generate"/>	
(Used as default value in certificate profiles using this CA)	

In the **Services** section, press **Generate Button** in **Default OCSP Service Locator**. Change also the localhost parameter by the name of the host.

Services	
Default OCSP Service Locator [?]	http://edpsca:8080/ejbca/publicweb/status/ocsp
<input type="button" value="Generate"/>	
(Used as default value in certificate profiles using this CA)	

Click **Create** to create the EDPSADMIN subordinated CA.

Repeat the same steps to create the EDPSNQC CA, but giving it the subject DN to CN=EDPSNQC, O=EDPS.

The basic Certification Authority structure has been created.

Manage Certification Authorities

List of Certification Authorities

EDPSADMIN, (Active)
EDPSNQC, (Active)
EDPSROOT, (Active)
ManagementCA, (Active)

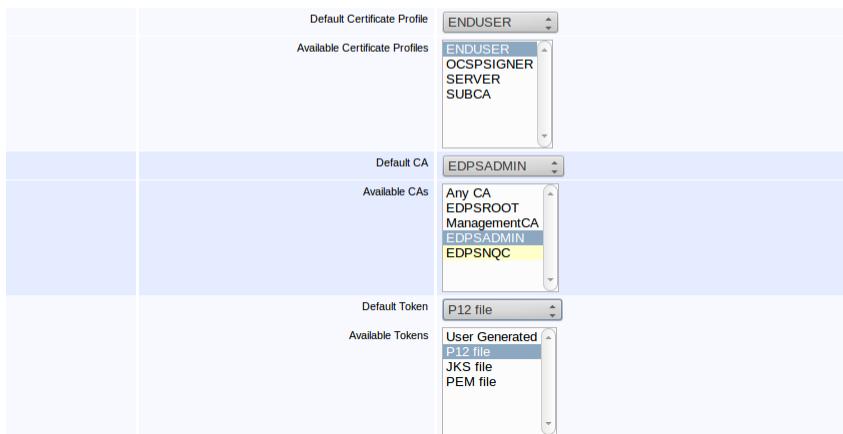
Set-up EJBCA to use SSL certificates from EDPSADMIN SUBCA

The next steps should be done to perform this task:

- Creating a new SuperAdministrator SSL Client certificate issued by the EDPSADMIN subordinated CA and installing it into the browser.
- Giving SuperAdministrator permissions to the SuperAdministrator SSL Client certificate.
- Creating the server SSL certificate.
- Configuring JBOSS to use this server SSL certificate and to accept SSL client certificates from the EDPSADMIN CA.

Create a Super Administrator certificate and set permissions

Create an End Entity Profile called **ADMIN_SUPERADMIN_EEPROFILE** with fixed **O=EDPS**, **OU=SuperAdministrator** in the subject. Choose **ENDUSER** in certificate profiles and allow the profile only to be issuable from the **EDPSADMIN** CA and token in **P12 File**.



Create user **superadmin1**, using the **ADMIN_SUPERADMIN_EEPROFILE**. Then, create and install the certificate in the browser by creating a PKCS#12 file through the public portal.

Add End Entity

End Entity Profile	ADMIN_SUPERADMIN_EEPROFILE	Required
Username	superadmin1	<input checked="" type="checkbox"/>
Password (or Enrollment Code)	<input checked="" type="checkbox"/>
Confirm Password	<input checked="" type="checkbox"/>
Subject DN Attributes		
CN, Common name	SuperAdmin1	<input checked="" type="checkbox"/>
O, Organization	EDPS	<input checked="" type="checkbox"/>
OU, Organizational Unit	SUPERADMIN	<input checked="" type="checkbox"/>
Main certificate data		
Certificate Profile	ENDUSER	<input checked="" type="checkbox"/>
CA	EDPSADMIN	<input checked="" type="checkbox"/>
Token	P12 file	<input checked="" type="checkbox"/>
		<input type="button" value="Add"/> <input type="button" value="Reset"/>

EJBCA Token Certificate Enrollment

Welcome to keystore enrollment.

If you want to, you can manually install the CA certificate retrieved.

Install CA certificates:

[Certificate chain](#)

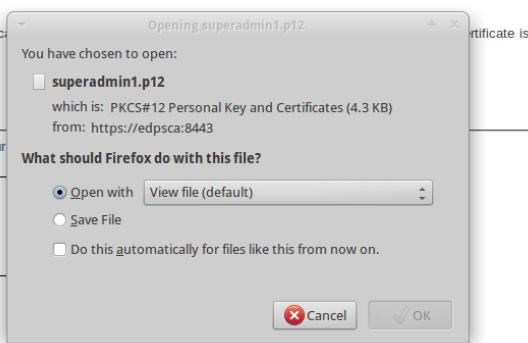
Please choose a key length, then click OK to fetch your certificate.

Options

Leave values as default if unsure.

Key length: 2048 bits

[Enroll](#)



Go to the **Administrator Roles** menu option, and add **OU=SUPERADMIN** with the **EDPSADMIN** CA to the **Super Administrator Role** clicking the **Administrators** label.

Edit Administrators

Administrator Role : Super Administrator Role

CA	Match with	Match type	Match value	
EDPSADMIN	X509: OU, Organizational Unit	Equal, case sens.	SuperAdministrator	Add
ManagementCA	X509:CN, Common name	Equal, case sens.	Super/Admin	Delete
EDPSADMIN	X509:OU, Organizational Unit	Equal, case sens.	Super/Administrator	Delete
CLI:Username		Equal, case sens.	ejbca	Delete

Made by PrimeKey Solutions AB, 2002–2013.

Create a new JBOSS SSL server certificate and replace the existing one

Create an End Entity profile called **ADMIN_SSLSERVER_EEPROFILE**, using the **SERVER** certificate profile, attached to the **EDPSADMIN** CA, and with token as **JKS** file.

Main certificate data	
Default Certificate Profile	SERVER
Available Certificate Profiles	ENDUSER OCSPSIGNER SERVER SUBCA
Default CA	EDPSADMIN
Available CAs	Any CA EDPSROOT ManagementCA EDPSADMIN EDPSNQC
Default Token	JKS file
Available Tokens	User Generated P12 file JKS file PEM file

Then, create the End Entity with this profile, entering the hostname (mandatory SSL requirements) in the CN.

Add End Entity

End Entity Profile	ADMIN_SSLSERVER_EEPROFILE	Required
Username	edpsca	<input checked="" type="checkbox"/>
Password (or Enrollment Code)	***	<input checked="" type="checkbox"/>
Confirm Password	***	<input checked="" type="checkbox"/>
Subject DN Attributes		
CN, Common name	edpsca	<input checked="" type="checkbox"/>
Main certificate data		
Certificate Profile	SERVER	<input checked="" type="checkbox"/>
CA	EDPSADMIN	<input checked="" type="checkbox"/>
Token	JKS file	<input checked="" type="checkbox"/>
<input type="button" value="Add"/> <input type="button" value="Reset"/>		

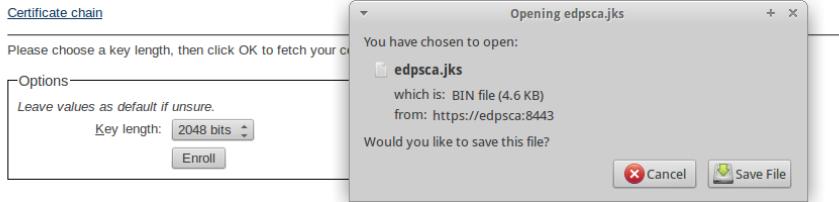
Go to the public portal and select **Create KeyStore**. Generate the certificate and download the .jks file. This file, like PKCS#12 files, contains the certificate and the private key. This is called the server keystore.

EJBCA Token Certificate Enrollment

Welcome to keystore enrollment.

If you want to, you can manually install the CA certificate(s) in your browser, otherwise this will be done automatically when your certificate is retrieved.

Install CA certificates:



We also need to instruct JBOSS HTTP/S server on what certificates can be used for SSL authentication (the so called truststore): go to **Fetch CA Certificates** (on the public web) and download the EDPSADMIN CA certificate chain called **Download JKS truststore**.

CA: ManagementCA

CN=ManagementCA,O=EDPS

CA certificate: [Download as PEM](#), [Download to Firefox](#), [Download to Internet Explorer](#)

CA certificate chain: [Download PEM chain](#), [Download JKS](#)

CA: EDPSADMIN

CN=EDPSROOT,O=EDPS

CA certificate: [Download as PEM](#), [Download to Firefox](#), [Download to Internet Explorer](#)

CN=EDPSADMIN,O=EDPS

CA certificate: [Download as PEM](#), [Download to Firefox](#)

CA certificate chain: [Download PEM chain](#), [Download JKS truststore](#) (password changeit)

CA: EDPSNQC

CN=EDPSROOT,O=EDPS

CA certificate: [Download as PEM](#), [Download to Firefox](#), [Download to Internet Explorer](#)

CN=EDPSNQC,O=EDPS

Stop the JBOSS server by pressing Control-C. Now, we'll create a back-up of the JBOSS folder.

Copy **edpsca.jks** (or the keystore file) to **/opt/jboss-as-7.1.1.Final/standalone/configuration/keystore/keystore.jks**

Copy **EDPSADMIN-chain.jks** (or the truststore file) to **/opt/jboss-as-7.1.1.Final/standalone/configuration/keystore/truststore.jks**

Edit the `/opt/jboss-as-7.1.1.Final/standalone/configuration/standalone.xml` file to match the passwords and the aliases of the keystore and truststore files.

Sometimes there are some problems with keystores. To avoid them, keep this reference commands in mind:

- Change password in a keystore: `keytool -storepasswd -keystore keystore.jks`
- List aliases in a keystore: `keytool -list -keystore keystore.jks`
- Change the alias of a keystore: `keytool -changealias -alias "oldalias" -destalias "newalias" -keystore keystore.jks -storepass xxxx`

Restart the JBOSS server by running again `./standalone.sh`

Check if you can logon with the new SuperAdministrator certificate.

Revoke the old management CA

Go to the **Certification Authorities** menu, select **Management CA** and click the **Edit CA** button.

Manage Certification Authorities

List of Certification Authorities

EDPSADMIN, (Active)
EDPSNQC, (Active)
EDPSROOT, (Active)
ManagementCA, (Active)

[Edit CA](#) [Delete CA](#) [Import CA keystore...](#) [Import CA certificate...](#)

Then, select a revocation reason like **Cessation of operation** and click the **Revoke** Button.



Go to **Administration Roles** and remove the Management CA as a source of **Super Administration Roles**.

By default, this CA is used to sign OCSP requests querying information from unknown CAs. Change now the configuration of OCSP. It will become active later in the redeployment process.

Copy or rename `ocsp.properties.sample` to `ocsp.properties` in EJBCA's `conf` folder.

Edit the `ocsp.properties` file and set

```
ocsp.defaultresponder=CN=EDPSROOT,O=EDPS
```

Create EDPSNQC CA administrators

Create the **ADMIN_CADMIN_EEPROFILE** profile following the same procedure than for **ADMIN_SUPERADMIN_EEPROFILE**, but with the fixed DN to **OU=CAADMIN** and **O=EDPS**.

Go to the **Administrator Roles** menu, and set **OU=CAADMIN** as a role of a new **Administrator Role** called **NQC CA Administrator**.

Edit Administrators

Administrator Role : NQC CA Administrator

CA		Match with	Match type	Match value		
EDPSADMIN		X509: OU, Organizational Unit	Equal, case sens.	CAADMIN	<input type="button" value="Add"/>	<input type="button" value="Edit Access Rules"/>
EDPSADMIN		X509:OU, Organizational Unit		CAADMIN		<input type="button" value="Delete"/>
Made by PrimeKey Solutions AB, 2002–2013.						

Set the permissions by clicking on **Edit Access Rules**. Select **CA Administrators** in **Role Template**, and select **EDPSNQC** in **Authorized CAs**.

Edit Access Rules

Administrator Role : NQC CA Administrator

						<input type="button" value="Back to Administrator roles"/>	<input type="button" value="Edit Administrators"/>
						<input type="button" value="Advanced Mode"/>	
Role Template		CA Administrators					
Authorized CAs		EDPSADMIN EDPSNQC EDPSROOT ManagementCA All					

Create the **caadmin1** user.

Add End Entity

End Entity Profile	ADMIN_CADMIN_EEPROFILE	Required
Username	caadmin1	<input checked="" type="checkbox"/>
Password (or Enrollment Code)	****	<input checked="" type="checkbox"/>
Confirm Password	****	<input checked="" type="checkbox"/>
Subject DN Attributes		
CN, Common name	CAAdmin1	<input checked="" type="checkbox"/>
O, Organization	EDPS	<input checked="" type="checkbox"/>
OU, Organizational Unit	CAADMIN	<input checked="" type="checkbox"/>
Main certificate data		
Certificate Profile	ENDUSER	<input checked="" type="checkbox"/>
CA	EDPSADMIN	<input checked="" type="checkbox"/>
Token	P12 file	<input checked="" type="checkbox"/>
<input type="button" value="Add"/> <input type="button" value="Reset"/>		

Made by PrimeKey Solutions AB, 2002–2013.

Go to the public portal, generate, download and install the PKCS#12 file with its certificate and private key.

Create NQC certificate templates and profiles

Log in into the Administration portal using the previously created Administrator certificate. As you can see, the functionality is reduced to the administration of the EDPSNQC CA.

Version : EJBCA 6.0.3 (r18324)

Welcome CAAdmin1 to EJBCA Administration.

Node hostname : edpsca
Server time : 2014-01-21 10:04:47+00:00

CA health state [?]		Publish queue status [?]	
CA Name	CA Service	CRL Status	Publisher
EDPSNQC			No publishers defined.

Made by PrimeKey Solutions AB, 2002–2013.

Home

CA Functions

- CA Activation
- CA Structure & CRLs
- Certificate Profiles

RA Functions

- Add End Entity
- End Entity Profiles
- Search End Entities
- User Data Sources

Supervision Functions

- Approve Actions
- View Log

System Functions

- Administrator Roles
- My Preferences

Public Web

- Documentation
- Logout

Create the certificate profile **EDPSNQC_1_3_130_2_1_CERTPROFILE**, with activated **CRL Distribution Points** and **OCSP/Authority Information Access**. In this case, we added the **Certificate Profile extension** to 1.3.130.2.1 to identify this type of certificate.

CRL Distribution Points [?]		<input checked="" type="checkbox"/> Use <input type="checkbox"/> Critical
Use CA defined CRL Dist. Point		<input checked="" type="checkbox"/> Use
CRL Distribution Point URI		<input type="text"/>
CRL Issuer [?]		<input type="text"/>
FreshestCRL extension [?]		<input type="checkbox"/> Use
Use CA Defined FreshestCRL extension		<input type="checkbox"/> Use
FreshestCRL extension URI		<input type="text"/>
Certificate Policies		<input checked="" type="checkbox"/> Use <input type="checkbox"/> Critical
<input type="button" value="Add"/>	Certificate Policy Id	<input type="text" value="1.3.130.2.1"/>
User Notice Text		<input type="text" value="This is a test certificate"/>
CPS URI		<input type="text"/>
Authority Information Access		<input checked="" type="checkbox"/> Use
Use CA defined OCSP locator		<input checked="" type="checkbox"/> Use

Create the certificate profile **EDPSNQC_1_3_130_2_2_CERTPROFILE**, with the same data, but using **1.3.130.2.2** as **Certificate Policy OID**, and “This is a production certificate” as **User Notice Text**. In this case, we have added that one extra approval is required to create this type of user for 1.3.130.2.2 certificates by setting the **Approval Settings** to **Add/Edit End Entity** and **Number of Required Approvals** to 1.

Approval Settings	<input type="checkbox"/> Add/Edit End Entity <input type="checkbox"/> Key Recovery <input type="checkbox"/> Revocation <input type="checkbox"/> CA Service Activation
Number of Required Approvals	<input type="text" value="1"/>

Create the end entity profile **EDPSNQC_1_3_130_2_1_EEPROFILE** by checking the **Autogenerated** password checkbox.

Edit End Entity Profile

End Entity Profile : **EDPSNQC_1_3_130_2_1_EEPROFILE**

	End Entity Profile Id	115026023	Back to End Entity Profiles
	Username	<input type="text"/>	
	Password (or Enrollment Code) [?]	<input type="text"/>	<input type="checkbox"/> Required <input checked="" type="checkbox"/> Autogenerated
	Digits only	<input type="text"/>	of length <input type="text" value="8"/>

Select **EDPSNQC_1_3_130_2_1_CERTPROFILE** and **User Generated** token.

	Default Certificate Profile	EDPSNQC_1_3_130_2_1_CERTPROFILE
	Available Certificate Profiles	EDPSNQC_1_3_130_2_1_CERTPROFILE EDPSNQC_1_3_130_2_2_CERTPROFILE ENDUSER OCSPSIGNER SERVER SUBCA
	Default CA	EDPSNQC
	Available CAs	EDPSNQC
	Default Token	User Generated
	Available Tokens	User Generated P12 file JKS file PEM file

We instruct EJBCA to add OU=NQC, O=EPDS to all issued certificates using this End Entity profile.

Subject DN Attributes [?]	
Select for Removal	Subject DN Attributes
<input type="checkbox"/>	CN, Common name <input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Modifiable
<input type="checkbox"/>	OU, Organizational Unit <input checked="" type="checkbox"/> Required <input type="checkbox"/> Modifiable
<input type="checkbox"/>	O, Organization <input checked="" type="checkbox"/> Required <input type="checkbox"/> Modifiable
<input type="button" value="Remove"/>	

As these kind of certificate users are self-enrollable, we need to check if the email address is correct. Thus, we instruct, via the End Entity template, that an email containing the password is sent to the user in order to allow him to log into the public portal and create the certificate. Check **Send Notification** and write the email text, using the \${USERNAME} and \${PASSWORD} substitution variables in the **Notification Message**.

<input type="button" value="Delete all"/>	<input style="border: none; padding: 0; margin: 0;" type="button" value="Send Notification [?]"/>	<input checked="" type="checkbox"/> Use : Default = <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Required
<input type="button" value="Delete"/>	Notification Sender	TheCATeam
	Notification Recipient	USER
	Notification Events	STATUSNEW STATUSFAILED STATUSINITIALIZED STATUSINPROCESS STATUSGENERATED STATUSREVOKED STATUSHISTORICAL
	Notification Subject	Your request is ready
	Notification Message	Go to CA web as use the following data: User: \${USERNAME} Password: \${PASSWORD}
<input type="button" value="Add"/>	Notification Sender	
	Notification Recipient	USER

Create the end entity profile **EDPSNQC_1_3_130_2_2_EEPROFILE**, without sending an email, but with more information attached to the user subject DN like **initials**, **surname** and **UID**.

Select for Removal	Subject DN Attributes	emailAddress, E-mail address in DN	Add
<input type="checkbox"/>	CN, Common name	<input type="text"/>	<input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Modifiable
<input type="checkbox"/>	initials, First name abbreviation	<input type="text"/>	<input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Modifiable
<input type="checkbox"/>	surname, Surname (last name)	<input type="text"/>	<input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Modifiable
<input type="checkbox"/>	UID, Unique Identifier	<input type="text"/>	<input checked="" type="checkbox"/> Required <input checked="" type="checkbox"/> Modifiable
<input type="checkbox"/>	O, Organization	<input type="text"/> EDPS	<input checked="" type="checkbox"/> Required <input type="checkbox"/> Modifiable
<input type="checkbox"/>	OU, Organizational Unit	<input type="text"/> NQC	<input checked="" type="checkbox"/> Required <input type="checkbox"/> Modifiable
<input type="button" value="Remove"/>			

Configure the website to enable self-registering to the portal and to send emails

To allow users to self-register with the End Entity profile **EDPSNQC_1_3_130_2_1_EEPROFILE** we need to edit **web.properties** in the **ejbca/conf** file and set:

```
web.selfreg.enabled=true
web.selfreg.defaultcerttype=1
web.selfreg.certtypes.1.description=Autoenrolled Test Certificate
web.selfreg.certtypes.1.eeprofile=EDPSNQC_1_3_130_2_1_EEPROFILE
web.selfreg.certtypes.1.certprofile=EDPSNQC_1_3_130_2_1_CERTPROFILE
```

For email configuration configure the **mail.properties** file in EJBCA's **conf** folder. Here's the configuration for a test Gmail account:

```
mail.user = ejbca.user.1
mail.password =edpsedps
mail.smtp.host=smtp.gmail.com
mail.smtp.port=465
mail.smtp.auth=true
mail.smtp.starttls.enable=true
mail.debug=true
```

Now we need to redeploy the EAR without installing the services again into the JBOSS. To do so, use the next instructions within the **ejbca** folder:

```
ant clean
ant deployear
ant jee:deployServices
```

If this doesn't work consider stopping and restarting the JBOSS server.

Creating a self-registered 1.3.130.2.1 user

Go to the public portal and select the **Request Registration** menu option. Press **Continue**.

Request Registration

Please enter your information below. A request for approval will be sent to your administrator.

Registration request - Step 1 of 2

Certificate type

Enter the data to autoenroll the user, in this case **ejbcuser1**. Then, press the **Request Registration** button.

Request Registration

Please enter your information below. A request for approval will be sent to your administrator.

Registration request - Step 2 of 2

Certificate type: Autoenrolled Test certificate

Name *

organizationalunit *

Organization *

Certificate retrieval

E-mail *

An auto-generated password will be sent to this e-mail address once the request has been approved.

Prevention of automatic registration (CAPTCHA)

Last character in e-mail *

* = Required field.

The next message will be shown because the registration must be approved by an administrator:

Request Registration

The registration request has been successfully submitted for approval by an administrator.

Enter the private administration portal with the CA Administration certificate, and go to the **Approve Actions** menu. The request will be shown. Click on it.

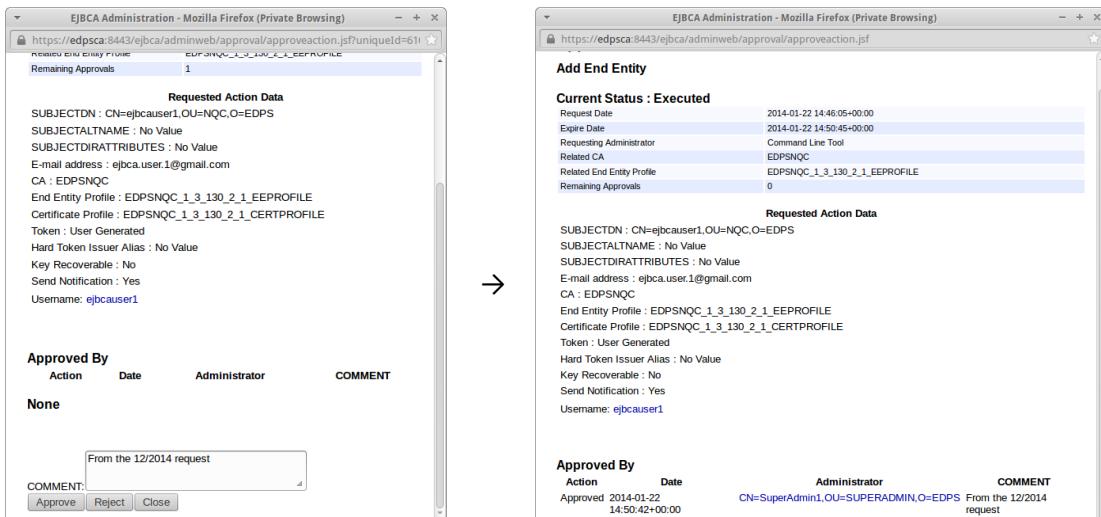
Approve Actions

Search for action with status requested within

Request Date	Approve Action Name	Requesting Administrator	Status
2014-01-22 14:03:42+00:00	Add End Entity	Command Line Tool	Waiting

1 Approval requests found.

A new window will appear. Enter a **COMMENT** and press the **Approve** button to approve the End Entity user creation action.



The screenshot shows two browser windows side-by-side. Both windows have the URL <https://edpsca:8443/ejbcn/adminweb/approveaction.jsp?uniqueId=61>.

Left Window (Initial Approval):

- Requested Action Data:**
 - SUBJECTDN : CN=ejbcauser1,OU=NQC,O=EDPS
 - SUBJECTALNAME : No Value
 - SUBJECTDIRATTRIBUTES : No Value
 - E-mail address : ejbca.user1@gmail.com
 - CA : EDPSNQC
 - End Entity Profile : EDPSNQC_1_3_130_2_1_EEPROFILE
 - Certificate Profile : EDPSNQC_1_3_130_2_1_CERTPROFILE
 - Token : User Generated
 - Hard Token Issuer Alias : No Value
 - Key Recoverable : No
 - Send Notification : Yes
 - Username: [ejbcauser1](#)
- Approved By:**

Action	Date	Administrator	Comment
None			

From the 12/2014 request

COMMENT:

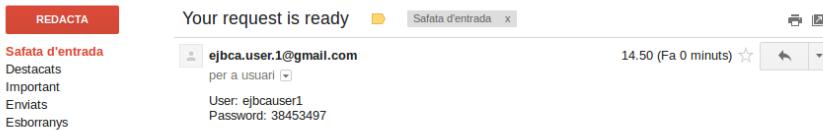
Buttons: Approve, Reject, Close

Right Window (After Approval):

- Current Status : Executed**
 - Request Date: 2014-01-22 14:46:05+00:00
 - Expire Date: 2014-01-22 14:50:45+00:00
 - Requesting Administrator: Command Line Tool
 - Related CA: EDPSNQC
 - Related End Entity Profile: EDPSNQC_1_3_130_2_1_EEPROFILE
 - Remaining Approvals: 0
- Requested Action Data:** (Same as Left Window)
- Approved By:**

Action	Date	Administrator	Comment
Approved	2014-01-22 14:50:42+00:00	CN=SuperAdmin1,OU=SUPERADMIN,O=EDPS	From the 12/2014 request

When the user is created, the email trigger specified in the End Entity profile is fired, so an email is sent to the user.



The screenshot shows an email inbox with the following details:

- REDACTA**
- Your request is ready**
- Safata d'entrada**
- ejbca.user.1@gmail.com**
- per a usuari**
- 14.50 (Fa 0 minutes)**
- Destacats**
- Important**
- Envíats**
- Esborranyos**
- User: ejbcauser1**
- Password: 38453497**

This password can be used to enroll into the public portal and create the certificate.

Certificate Enrollment

Welcome to Certificate Enrollment.

Please enter your username and enrollment code. Then click OK to generate your token.

Authentication

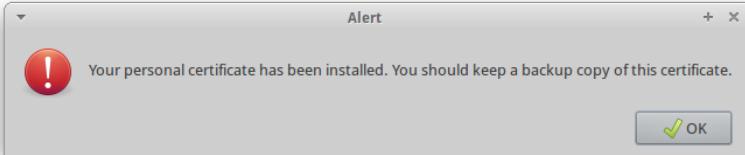
Username: <input type="text" value="ejbcauser1"/>
Enrollment code: <input type="text" value="*****"/>
<input type="button" value="OK"/>

Certificate Created

Subject DN: CN=ejbcauser1,OU=NQC,O=EDPS
 Issuer DN: CN=EDPSNQC,O=EDPS
 Serial Number: 35D206E7F6DF026A

If your certificate is not installed automatically, please click here to install it:

[Install certificate](#)



The alert dialog box contains the message: "Your personal certificate has been installed. You should keep a backup copy of this certificate." It features a red exclamation mark icon and an "OK" button with a checkmark.

Repeat the operation to create the ejbcauser2 with CN=ejbcauser2 with EDPSNQC_1_3_130_2_1_EEPROFILE.

Registering the users as RA administrators

Once the CN=ejbcuser1 and CN=ejbcuser2 certificates are created, they must be authorized as RA Administrators.

Manage Administrator Roles[?]

Roles

RA Administrator	Administrators	Access Rules	Rename	Delete
----------------------------------	--------------------------------	------------------------------	------------------------	------------------------

[Add](#)

Made by PrimeKey Solutions AB, 2002–2013.

Go to **Manage Administrator Roles** and create a new role named **RA Administrator**. Set the serial numbers of the certificates in the role filter setting:

Edit Administrators

Administrator Role : RA Administrator

Administrator Role : RA Administrator				
Back to Administrator roles Edit Access Rules				
CA	Match with	Match type	Match value	
EDPSNQC	X509: Certificate serial number (Recommended)	Equal, case sens.	2EB914DA2C503DD5	Add
EDPSNQC	X509: Certificate serial number (Recommended)	Equal, case sens.	35D206E7F6DF026A	Delete
EDPSNQC	X509: Certificate serial number (Recommended)	Equal, case sens.	35D206E7F6DF026A	Delete

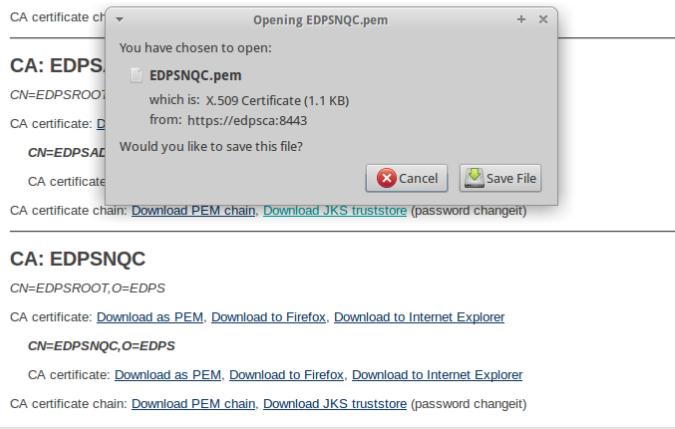
Go to **Edit Access Rules** and set the permissions for these RA Administrators.

Role Template		RA Administrators
Authorized CAs	EDPSNQC	
End Entity Rules	View End Entities View History Create End Entities Edit End Entities Delete End Entities Revoke End Entities Approve End Entities View Hard Token PUK Codes	
End Entity Profiles	All EDPSNQC_1_3_130_2_2_EEPROFILE EDPSNQC_1_3_130_2_1_EEPROFILE	

Configuring JBOSS to accept the RA administrator certificates as valid SSL client certificates

The JBOSS HTTP/S server is now configured to accept certificates from EDPSADMIN CA, but RA Administrator certificates come from EDPSNQC CA.

Go to the public portal and download the EDPSNQC certificate as a PEM file.



CA: EDPSNQC

CN=EDPSROOT, O=EDPS

CA certificate: [Download as PEM](#), [Download to Firefox](#), [Download to Internet Explorer](#)

CN=EDPSNQC, O=EDPS

CA certificate: [Download as PEM](#), [Download to Firefox](#), [Download to Internet Explorer](#)

CA certificate chain: [Download PEM chain](#), [Download JKS truststore](#) (password changeit)

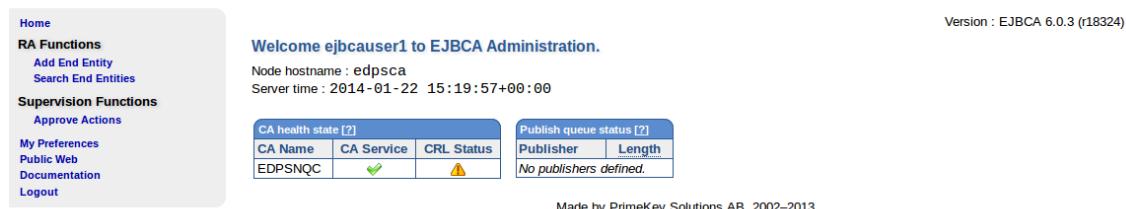
Then, import this certificate into the JBOSS truststore with this command:

```
user@edpsca:/opt/jboss-as-7.1.1.Final/standalone/configuration/keystore$ keytool -importcert -file EDPSNQC
.pem -alias nqc -keystore truststore.jks -storepass changeit
Certificate was added to keystore
```

Restart JBOSS in order to apply the changes.

Creating a 1.3.130.2.2 user with dual control (two approvals)

If we enter EJBCA with the CN=ejbcauser1 RA administrator SSL certificate, the following screen will be shown. As you can see, it is a reduced version of the CA administrator portal.



CA health state [?]			Publish queue status [?]	
CA Name	CA Service	CRL Status	Publisher	Length
EDPSNQC	✓	⚠	No publishers defined.	

Go to the **Add Entity** menu option, and fill out the data for the user with profile **EDPSNQC_1_3_130_2_2**. The entity will not be added automatically. Instead, a message will be shown saying that the **Request has been sent for approval**, as another approval is needed to add this user.

Add End Entity

Request has been sent for approval.

End Entity Profile	EDPSNQC_1_3_130_2_2_EEPROFILE	Required
--------------------	-------------------------------	----------

If the ejbcauser2 logs into EJBCA and goes to the **Approve Actions** menu, the request will be shown.

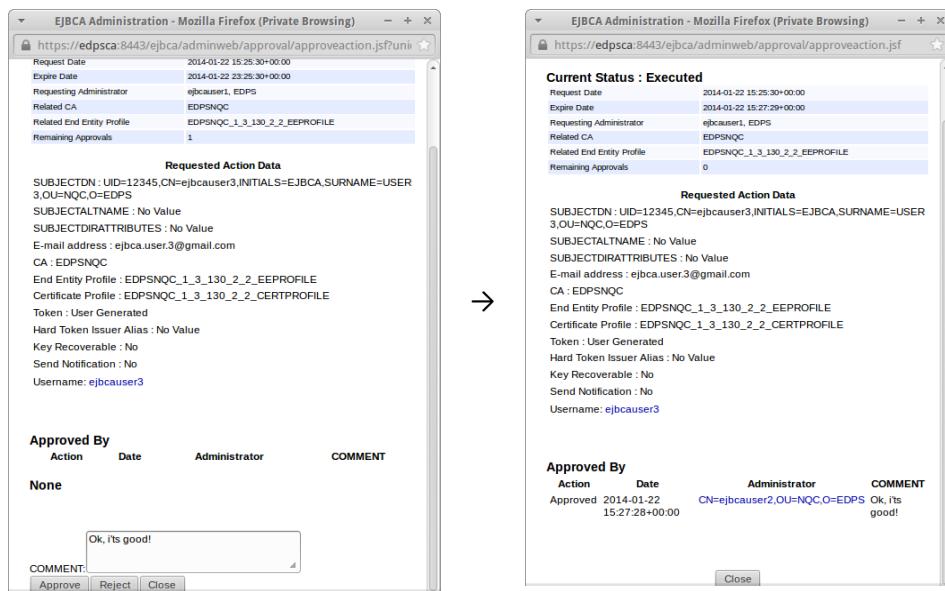
Approve Actions

Search for action with status: Waiting requested within: 30 minutes Search

Request Date	Approve Action Name	Requesting Administrator	Status
2014-01-22 15:25:30+00:00	Add End Entity	ejbcauser1, EDPS	Waiting

1 Approval requests found.

If you click the request entry, the approve action will be shown. Enter a message in the **COMMENT** text area and click the **Approve** button to create the user.



Requested Action Data

Request Date: 2014-01-22 15:25:30+00:00
 Expire Date: 2014-01-22 23:25:30+00:00
 Requesting Administrator: ejbcuser1, EDPS
 Related CA: EDPSNQC
 Related End Entity Profile: EDPSNQC_1_3_130_2_2_EEPROFILE
 Remaining Approvals: 1

SUBJECTDN: UID=12345,CN=ejbcuser3,INITIALS=EJBCA,SURNAME=USER3,OU=NQC,O=EDPS
 SUBJECTALTNAME: No Value
 SUBJECTDIRATTRIBUTES: No Value
 E-mail address: ejbca.user.3@gmail.com
 CA: EDPSNQC
 End Entity Profile: EDPSNQC_1_3_130_2_2_EEPROFILE
 Certificate Profile: EDPSNQC_1_3_130_2_2_CERTPROFILE
 Token: User Generated
 Hard Token Issuer Alias: No Value
 Key Recoverable: No
 Send Notification: No
 Username: ejbcuser3

Approved By

Action	Date	Administrator	Comment
None			

COMMENT: Ok, it's good!

Buttons: Approve, Reject, Close

Current Status : Executed

Request Date: 2014-01-22 15:25:30+00:00
 Expire Date: 2014-01-22 15:27:29+00:00
 Requesting Administrator: ejbcuser1, EDPS
 Related CA: EDPSNQC
 Related End Entity Profile: EDPSNQC_1_3_130_2_2_EEPROFILE
 Remaining Approvals: 0

SUBJECTDN: UID=12345,CN=ejbcuser3,INITIALS=EJBCA,SURNAME=USER3,OU=NQC,O=EDPS
 SUBJECTALTNAME: No Value
 SUBJECTDIRATTRIBUTES: No Value
 E-mail address: ejbca.user.3@gmail.com
 CA: EDPSNQC
 End Entity Profile: EDPSNQC_1_3_130_2_2_EEPROFILE
 Certificate Profile: EDPSNQC_1_3_130_2_2_CERTPROFILE
 Token: User Generated
 Hard Token Issuer Alias: No Value
 Key Recoverable: No
 Send Notification: No
 Username: ejbcuser3

Approved By

Action	Date	Administrator	Comment
Approved	2014-01-22 15:27:28+00:00	CN=ejbcuser2,OU=NQC,O=EDPS	Ok, it's good!

Buttons: Close

Creating an EJBCA service to automatically update CRLs

By default, EJBCA does not renew the CRLs automatically, although it can be easily configured to do so.

Go to the **Services** menu in the Administration console, and add service **SERVICE_CRL_UPDATE**.

Manage Services

List of Services

--

Edit Service | Delete Service

Add Service

SERVICE_CRL_UPDATE | Add | Rename | Use selected as template

Edit it and set the following parameters. Don't forget to mark the service as **Active**.

Service : SERVICE_CRL_UPDATE

Select Worker [?]	CRL Updater	Update
CRL Update Worker Settings		
CAs to Check		
Any CA EDPSROOT ManagementCA EDPSADMIN EDPSNQC		
Select Interval	Periodical Interval	
Periodical Interval Settings	Period	5 minutes
Select Action	No Action	
General Settings		
Active <input checked="" type="checkbox"/> Active		

Publishing issued certificates in an LDAP directory

To publish the certificates in an LDAP directory the next steps are required:

- Installing an LDAP server, in this case OpenDS will be used
- Creating a publisher connected to the LDAP
- Configuring EJBCA CAs End Entity templates to specify the publisher to be used
- Creating a service to perform automatic publishing
- Republishing all certificates

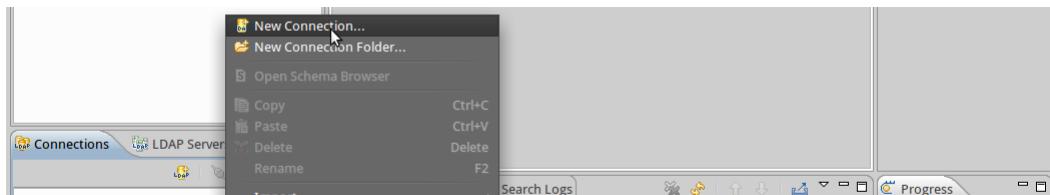
Install and configure OpenDS

Download and install ApacheDS 2.0.0-M15-64bit.bin. Follow the instructions from the README file.

Start the OpenDS service with:

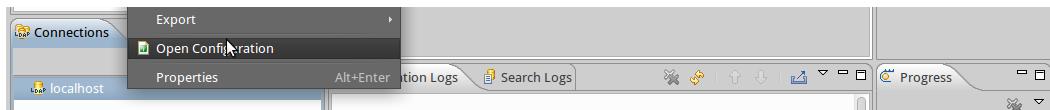
```
sudo /etc/init.d/apacheds-2.0.0-M15-default start
```

To configure ApacheDS, install the Apache Directory Studio, go to the **Connections** tab, left click and select **New connection...**

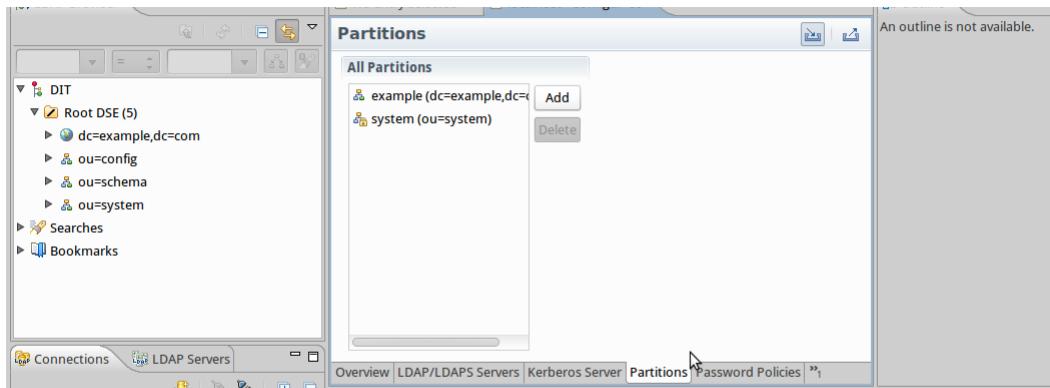


Set the **LDAP Hostname** to localhost, the **port** to 10389, the **Bind DN** to uid=admin, ou=system, and the **Bind password** to “secret”.

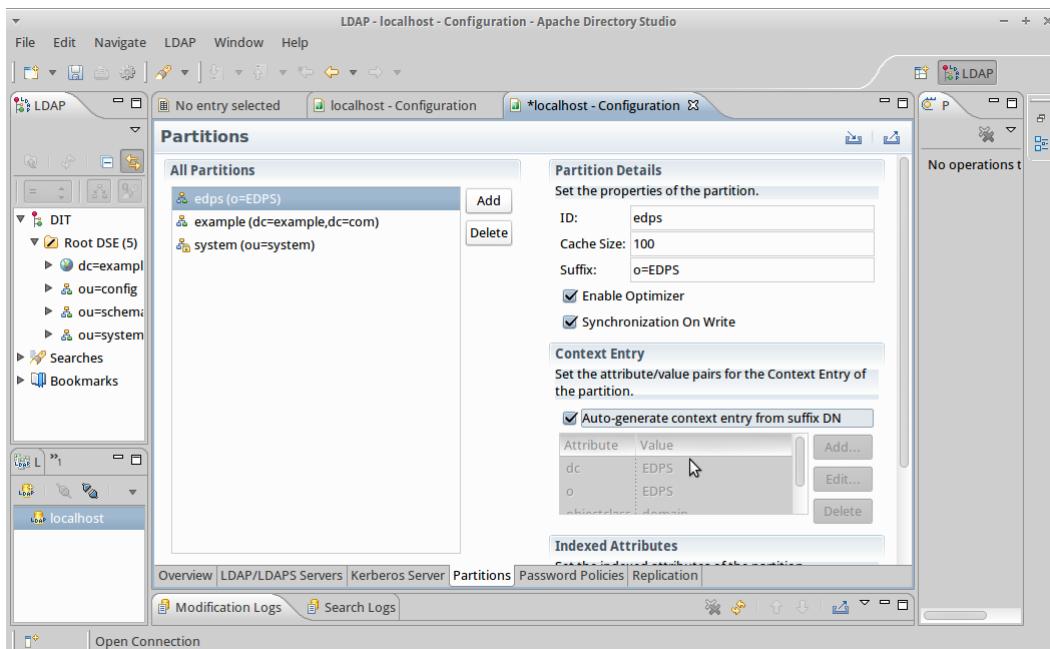
Select **Open Configuration** by right-clicking the **localhost** entry in the **Connections** tab.



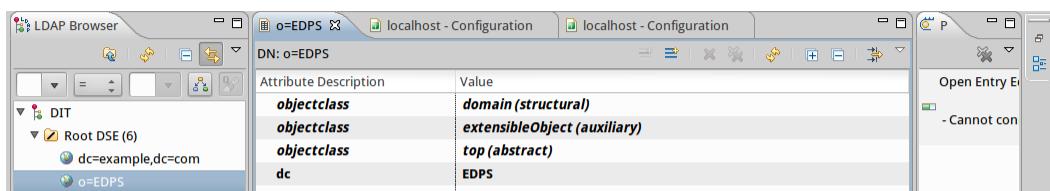
Select the **Root DSE** in the left tree. Select then the **Partitions** tab and click **Add**.



Set the suffix to **o=EDPS**.



Refresh the Root DSE node with the **Refresh** menu option by right-clicking the node. The new entry should be now available.



Configure EJBCA to publish certificates in the LDAP

Go to the **Publishers** menu in the administration portal and create a new publisher named **PUBLISHER_LDAP**.

List of Publishers

Edit Publisher	
Delete Publisher	

Add Publisher

PUBLISHER_LDAP	<input type="button" value="Add"/>	<input type="button" value="Rename"/>
Use selected as template		

Configure it with the same parameters introduced in the ApacheDS configuration.

Edit Publisher

Publisher : PUBLISHER_LDAP

		Back to Publishers
Name	PUBLISHER_LDAP	
Publisher Type	LDAP V3 Publisher	
LDAP Settings [?]		
Hostnames	localhost	
Port	10389	<input checked="" type="radio"/> Plain text Connection <input type="radio"/> STARTTLS Extension <input type="radio"/> TLS Connection
Base DN	o=EDPS	
	(appended to location fields to form a LDAP DN)	
Login DN	uid=admin, ou=system	
Login Password	*****	
Confirm Password	*****	

It is important to set the following options in the configuration:

LDAP operations	
Create Non-existing Users	<input checked="" type="checkbox"/> Activate
Modify Existing Users	<input checked="" type="checkbox"/> Activate
Overwrite Existing Attributes	<input type="checkbox"/> Activate
Add Non-existing Attributes	<input checked="" type="checkbox"/> Activate
Create intermediate nodes	<input checked="" type="checkbox"/> Activate
Add multiple certificates per user	<input checked="" type="checkbox"/> Activate
Remove certificates when revoked	<input checked="" type="checkbox"/> Activate
Remove Idap user when certificate revoked	<input checked="" type="checkbox"/> Activate
Set userPassword attribute	<input type="checkbox"/> Activate
LDAP location fields from certificate DN	emailAddress, E-mail address in DN UID, Unique Identifier CN, Common name serialNumber, Serial number (in DN) givenName, Given name (first name) initials, First name abbreviation surname, Surname (last name) title, Title OU, Organizational Unit O, Organization <small>(DC, O, ST and C fields should be defined in BaseDN)</small>

Create a new service called **SERVICE_PUBLISH_LDAP** in the **Services** main menu option.

Add Service

SERVICE_PUBLISH_LDAP	<input type="button" value="Add"/>	<input type="button" value="Rename"/>
<input type="button" value="Use selected as template"/>		

To configure the service select **Worker Publishing Queue Process Service**, and in **Publishers to check** the LDAP publisher **PUBLISHER_LDAP**.

Service : SERVICE_PUBLISH_LDAP

Select Worker [?]		Back to Services
Publish queue process settings		<input type="button" value="Update"/>
Publishers to check		PUBLISHER_LDAP (1888315867)
Select Interval		Periodical Interval
Periodical Interval Settings		Period: 5 minutes
Select Action		No Action
General Settings		Active <input checked="" type="checkbox"/> Active

Edit the **EDPSNQC CA**, **EDPSNQC_1_3_130_2_1_EEPROFILE** and **EDPSNQC_1_3_130_2_2_EEPROFILE** to set the publisher to **PUBLISHER_LDAP**.



Republishing all certificates in the LDAP

To republish all certificates in the LDAP, execute this command from the EJBCA folder.

```
bin/ejbca.sh ca republish EDPSNQC -all
```

If you go to Apache Directory Studio and refresh the entries, the users will be shown.

Attribute Description	Value
objectclass	inetOrgPerson (structural)
objectclass	organizationalPerson (structural)
objectclass	person (structural)
top	(abstract)
CN	ejbcauser2
sn	ejbcauser2
O	EDPS
ou	NO

Accessing the CA remotely

To access the CA remotely from the command line, go to the EJBCA folder and execute

```
ant clientToolBox
```

When compiled, move the **dist/clientToolBox** folder to the remote host.

Copy the administrator certificate in the form of a PKCS#12 (e.g. caadmin1.p12) to the remote host, too.

In the remote host, edit the **ejbcawsracli.properties** file and set

```
ejbcawsracli.url = https://edpsca:8443/ejbca/ejbcaws/ejbcaws
```

```
ejbcawsracli.keystore.path = caadmin1.p12
```

Now, you can query and operate remotely executing script **ejbcaClientToolBox.sh**

For instance, with this command you can see how many users in the CA contain the “adria” substring.

```
./ejbcaClientToolBox.sh EjbcawsRaCli finduser USERNAME CONTAINS adria
```