



|  |
| --- |
| X-Road: Central Server Configuration  **Data Model** |
| Version: 1.3  11.12.2015  41 pages  Doc. ID: DM-CS |

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 15.06.2015 | 0.1 | Initial version | Martin Lind |
| 30.06.2015 | 0.2 | Comments and revisions | Margus Freudenthal |
| 9.07.2015 | 0.3 | Rearrangements for consistency | Martin Lind |
| 28.08.2015 | 0.4 | Corrections according to feedback to first 7 tables. | Martin Lind |
| 1.09.2015 | 0.5 | Better explanations for modifications to all tables. | Martin Lind |
| 09.09.2015 | 0.6 | Made minor editorial changes | Margus Freudenthal |
| 16.09.2015 | 0.7 | Added the descriptions of the fields and procedures related to high availability | Marju Ignatjeva |
| 21.09.2015 | 1.0 | Editorial changes made | Imbi Nõgisto |
| 16.10.2015 | 1.1 | Field cert\_profile\_info for approved CA-s table and one missing index | Martin Lind |
| 17.10.2015 | 1.2 | Clarified description of the cert\_profile\_info field | Margus Freudenthal |
| 11.12.2015 | 1.3 | Subsystems can only be clients of security servers | Siim Annuk |

Table of Contents

[1 General 7](#_Toc438060017)

[1.1 Preamble 7](#_Toc438060018)

[1.2 Database Version 7](#_Toc438060019)

[1.3 Creating, Backing Up and Restoring the Database 7](#_Toc438060020)

[1.4 Saving Database History 8](#_Toc438060021)

[1.5 High Availability Support 8](#_Toc438060022)

[1.6 Entity-Relationship Diagram 8](#_Toc438060023)

[1.7 List of Stored Procedures 10](#_Toc438060024)

[1.8 List of Triggers 11](#_Toc438060025)

[2 Description of Entities 12](#_Toc438060026)

[2.1 ANCHOR\_URL\_CERTS 12](#_Toc438060027)

[2.1.1 Indexes 12](#_Toc438060028)

[2.1.2 Attributes 12](#_Toc438060029)

[2.2 ANCHOR\_URLS 13](#_Toc438060030)

[2.2.1 Indexes 13](#_Toc438060031)

[2.2.2 Attributes 13](#_Toc438060032)

[2.3 APPROVED\_CAS 14](#_Toc438060033)

[2.3.1 Indexes 14](#_Toc438060034)

[2.3.2 Attributes 14](#_Toc438060035)

[2.4 APPROVED\_TSAS 15](#_Toc438060036)

[2.4.1 Attributes 15](#_Toc438060037)

[2.5 AUTH\_CERTS 16](#_Toc438060038)

[2.5.1 Indexes 16](#_Toc438060039)

[2.5.2 Attributes 17](#_Toc438060040)

[2.6 CA\_INFOS 17](#_Toc438060041)

[2.6.1 Indexes 18](#_Toc438060042)

[2.6.2 Attributes 18](#_Toc438060043)

[2.7 CENTRAL\_SERVICES 18](#_Toc438060044)

[2.7.1 Indexes 19](#_Toc438060045)

[2.7.2 Attributes 19](#_Toc438060046)

[2.8 CONFIGURATION\_SIGNING\_KEYS 19](#_Toc438060047)

[2.8.1 Indexes 20](#_Toc438060048)

[2.8.2 Attributes 20](#_Toc438060049)

[2.9 CONFIGURATION\_SOURCES 20](#_Toc438060050)

[2.9.1 Indexes 21](#_Toc438060051)

[2.9.2 Attributes 21](#_Toc438060052)

[2.10 DISTRIBUTED\_FILES 22](#_Toc438060053)

[2.10.1 Attributes 22](#_Toc438060054)

[2.11 GLOBAL\_GROUP\_MEMBERS 23](#_Toc438060055)

[2.11.1 Indexes 23](#_Toc438060056)

[2.11.2 Attributes 23](#_Toc438060057)

[2.12 GLOBAL\_GROUPS 24](#_Toc438060058)

[2.12.1 Attributes 24](#_Toc438060059)

[2.13 HISTORY 25](#_Toc438060060)

[2.13.1 Attributes 25](#_Toc438060061)

[2.14 IDENTIFIERS 26](#_Toc438060062)

[2.14.1 Attributes 26](#_Toc438060063)

[2.15 MEMBER\_CLASSES 27](#_Toc438060064)

[2.15.1 Attributes 27](#_Toc438060065)

[2.16 OCSP\_INFOS 27](#_Toc438060066)

[2.16.1 Indexes 28](#_Toc438060067)

[2.16.2 Attributes 28](#_Toc438060068)

[2.17 REQUEST\_PROCESSINGS 28](#_Toc438060069)

[2.17.1 Attributes 29](#_Toc438060070)

[2.18 REQUESTS 29](#_Toc438060071)

[2.18.1 Indexes 30](#_Toc438060072)

[2.18.2 Attributes 31](#_Toc438060073)

[2.19 SCHEMA\_MIGRATIONS 32](#_Toc438060074)

[2.19.1 Indexes 32](#_Toc438060075)

[2.19.2 Attributes 33](#_Toc438060076)

[2.20 SECURITY\_CATEGORIES 33](#_Toc438060077)

[2.20.1 Attributes 33](#_Toc438060078)

[2.21 SECURITY\_SERVER\_CLIENT\_NAMES 33](#_Toc438060079)

[2.21.1 Indexes 34](#_Toc438060080)

[2.21.2 Attributes 34](#_Toc438060081)

[2.22 SECURITY\_SERVER\_CLIENTS 34](#_Toc438060082)

[2.22.1 Indexes 35](#_Toc438060083)

[2.22.2 Attributes 35](#_Toc438060084)

[2.23 SECURITY\_SERVERS 36](#_Toc438060085)

[2.23.1 Indexes 36](#_Toc438060086)

[2.23.2 Attributes 36](#_Toc438060087)

[2.24 SECURITY\_SERVERS\_SECURITY\_CATEGORIES 37](#_Toc438060088)

[2.24.1 Indexes 37](#_Toc438060089)

[2.24.2 Attributes 37](#_Toc438060090)

[2.25 SERVER\_CLIENTS 38](#_Toc438060091)

[2.25.1 Indexes 38](#_Toc438060092)

[2.25.2 Attributes 38](#_Toc438060093)

[2.26 SYSTEM\_PARAMETERS 38](#_Toc438060094)

[2.26.1 Attributes 40](#_Toc438060095)

[2.27 TRUSTED\_ANCHORS 40](#_Toc438060096)

[2.27.1 Attributes 40](#_Toc438060097)

[2.28 UI\_USERS 41](#_Toc438060098)

[2.28.1 Attributes 41](#_Toc438060099)

License

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/.

# General

## Preamble

This document describes the database model of the X-Road central server.

## Database Version

This database assumes PostgreSQL version 9.3 or 9.4 depending on whether the central server is deployed in a simple setup or in a cluster for achieving high availability (HA) (see section 1.5 for details). Ubuntu 14.04 default settings are used in simple setup, while a custom configuration is used in HA setup.

## Creating, Backing Up and Restoring the Database

This database is integrated into X-Road central server application. The database management functions are embedded into the application user interface.

The database, the database user and the data model is created by the application's installer. The database updates are packaged as application updates and are applied when the application is upgraded. From the technical point of view, the database structure is created and updated using Rails migrations (see [http://guides.rubyonrails.org/](http://guides.rubyonrails.org/v3.2.21/migrations.html)). The migration scripts can be found both in application source and in WAR file of central server's web based user interface (WAR – [Java Web Archive](https://docs.oracle.com/javaee/7/tutorial/packaging003.htm)).

Database backup functionality is built into the application. The backup operation can be invoked from the web-based user interface or from the command line. The backup contains dump of all the database structure and contents. When restoring the application, first the software is installed and then the configuration database is restored together with all the other necessary files. This produces a working central server.

## Saving Database History

This section describes the general mechanism for storing the history of the database tables. All the history-aware tables have an associated trigger *update\_history* that records all the modifications to data. All the tables of central database are history-aware, except for

* **history**,
* **distributed\_files** and
* **schema\_migrations**

When a row is created, updated or deleted in one of the history-aware tables, the trigger *update\_history* is activated and invokes the stored procedure *add\_history\_rows.* For each changed column, *add\_history\_rows* inserts a row into the history table. The details of the stored procedures are described in section 1.7.

## High Availability Support

High availability of the X-Road central server has been implemented using the BDR extension of the PostgreSQL database (<http://2ndquadrant.com/en/resources/bdr/>). The implementation is an asynchronous multi-master setup where the database changes to any node of the cluster are replicated to the rest of the nodes. Each node runs a separate instance of the UI of the central server and uses separate keys for signing configuration.

In order to support high availability (HA) setup of the X-Road central server, some database tables have the *ha\_node\_name* field. In an HA setup, the name of the node of the cluster that initiated the insertion of a given record, is stored in that field. In ordinary setup, a default value is used. In both cases, this is done at the level of stored procedures as described in section 1.7.

The logic of taking into account the value of *ha\_node\_name* where applicable, has been implemented at the application level.

Database history records are aware of the node name in an HA setup and are replicated just like other records. Thus each node contains the full history of database changes. Because replication events happen at a lower level than insertions of records, the replication of history records themselves does not trigger any subsequent insertions of history records on target nodes.

## Entity-Relationship Diagram

The data model is described in two entity relationship diagrams (ERD). The first diagram contains tables related to security servers and security server clients. The second diagram contains the rest of the tables.

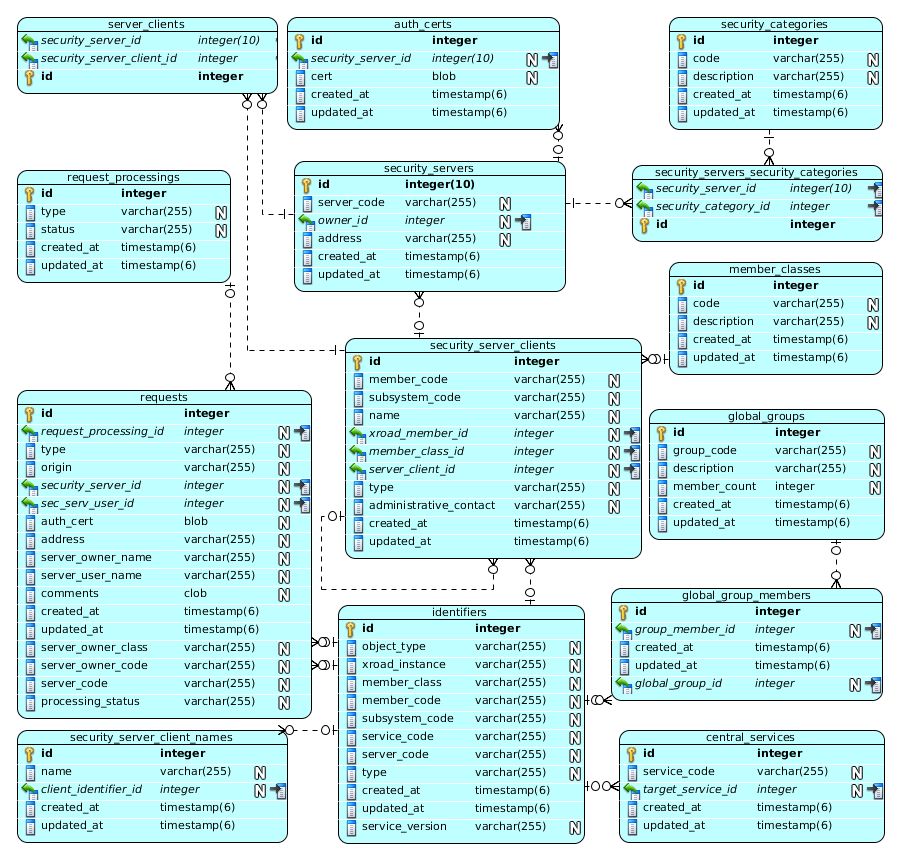
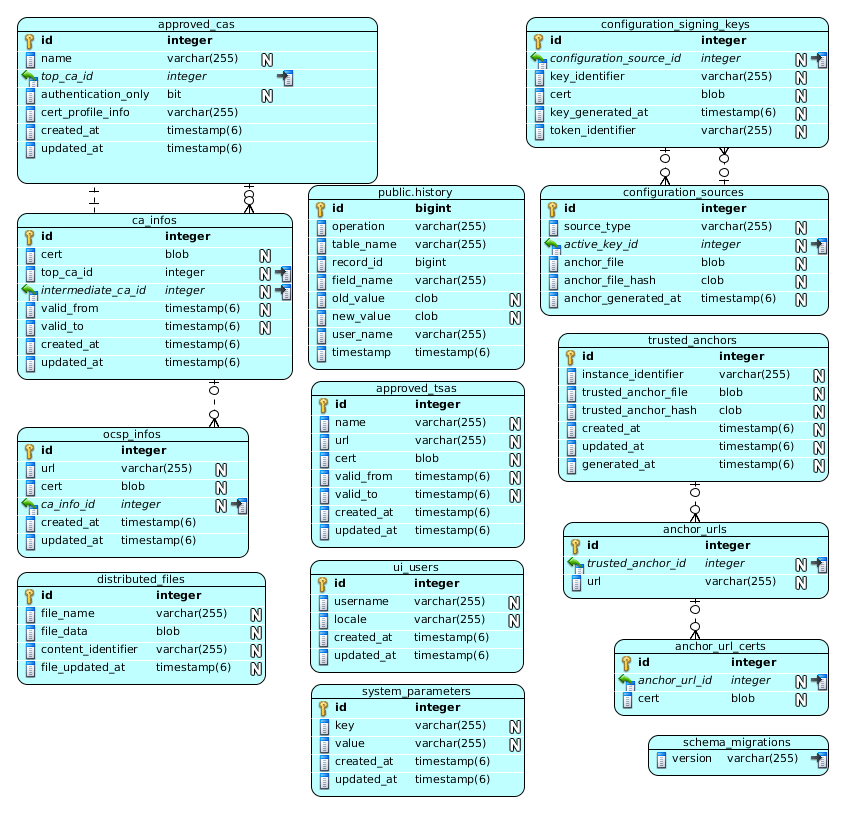


Figure 1. ERD describing security servers, security server clients and related database tables in the central server database

Figure 2. ERD describing entities not directly related to security servers and security server clients in the central server database

## List of Stored Procedures

The following stored procedures are present in the database, regardless of whether a given central server has been installed in standalone or HA setup.

1. **add\_history\_rows**: Detects the changes made as a result of the operation it is invoked on, and calls the **insert\_history\_row** stored procedure to insert a row to the history table, for each changed field. For insertions and deletions, a history record is inserted for each field of the original table.
2. **insert\_history\_row**: Inserts a single row with values corresponding to a changed field in one of the database tables. Invoked by the **add\_history\_rows** stored procedure.
3. **insert\_node\_name**: For each record inserted to a table with the field *ha\_node\_name*, sets the value of this field to

* the default value in standalone systems
* the name of the cluster node that initiated the insertion, in an HA setup.

## List of Triggers

The following triggers are present in the database, regardless of whether a given central server has been installed in standalone or HA setup.

1. **update\_history**: Invokes the **add\_history\_rows** stored procedure upon insertions, updates and deletions of records. Created for each history-aware table.
2. **insert\_node\_name**: Invokes the **insert\_node\_name** stored procedure upon insertions. Created for each table with the *ha\_node\_name* field.

# Description of Entities

## ANCHOR\_URL\_CERTS

Certificate belonging to a configuration source that is represented in database by an **anchor URL**. The certificates are used to verify signed configuration downloaded from a given URL.

The record is created when an X-Road security officer has received trusted anchor from federation partner and uploads it in the user interface. The record is deleted when the federation contract between two X-Road instances has come to an end and an X-Road security officer deletes the anchor associated with the record in the user interface. The record is never modified. Records in tables *trusted\_anchors* and *anchor\_urls* are created and deleted in exactly the same way. See also documentation of these tables.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_anchor\_url\_certs\_on\_anchor\_url\_id | anchor\_url\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| anchor\_url\_id [FK] | integer |  | ID of the configuration anchor URL the certificate belongs to. References **id** attribute of **anchor\_urls** entity. As every anchor URL certificate must belong to particular anchor URL, the column cannot be NULL (currently set in the data model layer of the user interface). |
| cert | bytea |  | Bytes of anchor URL certificate (in DER encoding, see also <http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf>). Cannot be NULL. |

## ANCHOR\_URLS

URL pointing to a configuration source that is described by a **trusted anchor**. Anchor URL is HTTP URL that can be used to download signed configuration.

The record is created or modified exactly the same way as described in the documentation of table *anchor\_url\_certs*. The record is never modified.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_anchor\_urls\_on\_trusted\_anchor\_id | trusted\_anchor\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| trusted\_anchor\_id [FK] | integer |  | ID of the trusted anchor that contains this anchor URL. References **id** attribute of **trusted\_anchors** entity. Cannot be NULL. |
| url | character varying(255) |  | The URL that can be used by the configuration client to download the configuration from the configuration source. Must correspond to the URL format (See also URL specification: <http://www.w3.org/Addressing/URL/url-spec.txt>). Cannot be NULL. |

## APPROVED\_CAS

Approved certification authority (CA) that is used when verifying authentication and signing certificates. Exactly one top-level CA certificate is associated with each approved CA. Multiple intermediate CA certificates can be associated with each approved CA. The intermediate CA certificates form a hierarchy with top-level CA used as a trust anchor. The intermediate CAs are used for certificate path building and for finding OCSP responders.

New record creation process starts when an X-Road system administrator receives a certificate from a CA which is going to be trusted by the X-Road instance. Having received the certificate, an X-Road system administrator uploads it in the user interface. The record is deleted when for any reason the governing authority of the X-Road instance does not trust the CA any more. Then an X-Road system administrator deletes the approved CA in the user interface. The record is modified when the user changes parameters of the approved CA. Parameters that can be changed, are following:

1. Flag “authentication only”, see also documentation of the column *authentication\_only* of this table.
2. Certificate profile info class name, see also documentation of the column *cert\_profile\_info* of this table.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_approved\_cas\_on\_top\_ca\_id | top\_ca\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| name | character varying(255) |  | Name of the CA, used in user interfaces. Technically this is the subject name of the top level certification authority certificate. |
| top\_ca\_id [FK] | integer |  | ID of the top level CA certificate entry of the record. See also documentation of the table *ca\_infos*. Cannot be NULL. |
| authentication\_only | boolean |  | If true, this CA can only issue authentication certificates. If false, this CA can issue all certificates. |
| cert\_profile\_info | character varying(255) |  | Fully qualified Java class name that implements the CertificateProfileInfoProvider interface. The implementing class is used for extracting subject information from certificates. The implementing class must be present in classpath of both central server and securitys servers. Cannot be NULL. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## APPROVED\_TSAS

Approved time-stamping authority (TSA). The certificate of the approved CA is used for time-stamping signed messages.

New record creation process starts when an X-Road system administrator receives a certificate from a TSA which is going to be trusted by the X‑Road instance. Having received the certificate, an X-Road system administrator uploads it in the user interface. The record is created when the user adds new approved TSA in the user interface. The record is deleted when for any reason the governing authority of the X-Road instance does not trust the TSA any more. Then an X-Road system administrator deletes the approved TSA in the user interface. The record is never modified.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| name | character varying(255) |  | Name of the TSA, used in user interfaces. Technically, this is the subject name of the TSA certificate. |
| url | character varying(255) |  | URL that is used for sending time-stamping requests. Must correspond to the URL format. Cannot be NULL. |
| cert | bytea |  | TSA certificate that is used to verify issued time stamps. Stored in DER-encoded form. Cannot be NULL. |
| valid\_from | timestamp without time zone |  | Start of validity period of the TSA's certificate. Extracted from the uploaded certificate. |
| valid\_to | timestamp without time zone |  | End of validity period of the TSA's certificate. Extracted from the uploaded certificate. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## AUTH\_CERTS

Authentication certificate that is used by a security server to establish secure connection. Each authentication certificate belongs to a particular security server.

Prerequisite for creating the record is that a pair of authentication certificate registration requests are submitted for approval (see also documentation of tables *requests* and *request\_processings*). The record is created whenever an X-Road registration officer approves one of the submitted requests in the user interface. The record is removed whenever there is need to remove the security server the record belongs to or when the authentication certificate cannot be used any more. An X-Road registration officer can either remove security server or send authentication certificate deletion request for the security server in the user interface. The latter is done when only authentication certificate (without security server) is going to be deleted. The record is never modified. See also documentation of table *security\_servers*.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_auth\_certs\_on\_security\_server\_id | security\_server\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| security\_server\_id [FK] | integer |  | ID of the security server the authentication certificate belongs to. References **id** attribute of **security\_servers** entity. Cannot be NULL. |
| cert | bytea |  | Authentication certificate contents (in DER encoding). Cannot be NULL. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## CA\_INFOS

CA certificates with additional data that is displayed in the user interface. The CA info can describe either certificate of a top-level CA or an intermediate CA. The record is created when a new top-level CA or an intermediate CA is added in the user interface.

The record is created on two occasions:

1. When a new approved CA is added in the user interface (for details, see documentation of table *approved\_cas*), CA info corresponding to top CA is added.
2. When the certification chain of the approved CA includes intermediate CA-s. Then an X-Road system administrator adds intermediate CA certificate(s) in the user interface.

Accordingly, the record is deleted when either the approved CA is deleted (see also documentation of table *approved\_cas*) or the intermediate CA is deleted in the user interface. The latter can happen when certification chain for approved CA changes. The record is never modified.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_ca\_infos\_on\_intermediate\_ca\_id | intermediate\_ca\_id |
| index\_ca\_infos\_on\_top\_ca\_id | top\_ca\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| cert | bytea |  | Contents of the CA certificate (in DER encoding). Cannot be NULL. |
| intermediate\_ca\_id [FK] | integer |  | Used to associate the *ca\_info* record with a top-level CA record. This field is present only for intermediate-level CAs (top-level CA is referenced directly by the *ca\_info* table. References to **id** attribute of **approved\_cas** entity. |
| valid\_from | timestamp without time zone |  | Start of validity period of the CA's certificate. Extracted from the certificate. |
| valid\_to | timestamp without time zone |  | End of validity period of the CA's certificate. Extracted from the certificate. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## CENTRAL\_SERVICES

Central service is a centrally defined and managed alias (service code) to a concrete service implemented by an X-Road member or a subsystem.

Central service record is created when there is a need for a new centrally defined service. Then an X-Road system administrator adds new central service in the user interface.

The record is modified when implementing service for the central service changes. Then an X-Road system administrator updates the target service in the user interface. The record is deleted when there is no need for the central service with this particular service code any more. Then an X-Road system administrator deletes the central service in the user interface.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_central\_services\_on\_target\_service\_id | target\_service\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| service\_code | character varying(255) |  | Code that uniquely identifies the central service in this X-Road instance. Cannot be NULL. |
| target\_service\_id [FK] | integer |  | ID of the implementing service identifier. Implementing service can be any X-Road service. References **id** attribute of **identifiers** entity. Can be NULL if there is no implementing service defined. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## CONFIGURATION\_SIGNING\_KEYS

Signing context (key identifier used by the signer and signing certificate) for signing the global configuration. A signing key belongs to a configuration source. A configuration signing key is used when it is marked as active in the user interface. Technically it is done by designating the key as active key in the *configuration\_sources* table, see also documentation of table *configuration\_sources*.

The record is created when a new key for signing global configuration is needed (either no keys are present or any of present ones cannot be used). Then an X-Road security officer generates a new signing key in the user interface. Non-active configuration signing keys that are no longer necessary can be deleted by an X-Road security officer in the user interface. The record is never modified.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_configuration\_signing\_keys\_on\_configuration\_source\_id | configuration\_source\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| configuration\_source\_id [FK] | integer |  | ID of the configuration source that uses this signing key. References **id** attribute of **configuration\_sources** entity. Cannot be NULL. |
| key\_identifier | character varying(255) |  | Used to identify this key in the Signer component that actually manages the key. Signing key identifiers for the same signing token must be unique. |
| cert | bytea |  | Contents of the configuration signing certificate (in DER encoding). |
| key\_generated\_at | timestamp without time zone |  | The signing key generation time. |
| token\_identifier | character varying(255) |  | Unique identifier of hardware or software token used for signing the configuration. |

## CONFIGURATION\_SOURCES

Configuration source that the central server uses to distribute the global configuration. Stores (with associated *configuration\_signing\_keys* table) all the data necessary to generate **configuration anchors** for the central server. The configuration distributed by the source can be either **internal configuration** or **external configuration**. The internal configuration is distributed to security servers of this X-Road instance. The external configuration is distributed to the other X-Road instances (federation partners).

The configuration source is associated with several configuration signing keys. When generating a configuration anchor for the source, all the keys are included. One of the keys is marked active. Technically, the active key (in c*onfiguration\_signing\_keys* table) is referred by the attribute *active\_key\_id*. The active key is used for signing configuration distributed by this source.

In an HA setup, each node of the cluster uses separate keys for signing configuration, and configuration anchors contain entries for each node of the cluster.

The record is created when the configuration source tab (either for internal or external configuration) is opened in the UI for the first time. The configuration source tab can be opened for viewing or editing configuration anchor or signing keys information for the configuration source. The record is modified when signing keys information of the configuration source is changed and a new configuration anchor is generated by the system. Also, the record is modified when an X-Road security officer generates a new configuration anchor in the user interface. The record is never deleted.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_configuration\_sources\_on\_active\_key\_id | active\_key\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| source\_type | character varying(255) |  | Type of the configuration source, can be either 'internal' or 'external'. |
| active\_key\_id [FK] | integer |  | ID of the active key that is used to sign the distributed configuration (all the other keys are only included in the generated configuration anchor).References **id** attribute of **configuration\_signing\_keys** entity. |
| anchor\_file | bytea |  | Configuration anchor file (in XML format). The anchor is re-generated if any information contained in the anchor is saved. |
| anchor\_file\_hash | text |  | Configuration anchor file hash (for displaying in user interface). Updated when the configuration anchor is re-generated. |
| anchor\_generated\_at | timestamp without time zone |  | Configuration anchor generation time. Updated when the configuration anchor is re-generated. |
| ha\_node\_name | character varying(255) |  | Name of the cluster node that initiated the insertion in an HA setup; the default value in standalone setup. |

## DISTRIBUTED\_FILES

Stores global configuration files that are distributed to the X-Road members. There are three kinds of distributed files:

* private parameters (distributed to only members of this X-Road instance),
* shared parameters (distributed to members of this X-Road instance and to members of federation partners),
* other configuration files (optional, depends on the configuration of the instance. The supported optional configuration files are described in the system configuration).

The record can be created in two different ways:

* The record corresponding to either private or shared parameters is created when a new global configuration is generated. Global configuration is triggered periodically (every minute) by a cron job.
* The record corresponding to other configuration file is created when there is a need to distribute new version of a configuration file specific to the X-Road instance. Then an X-Road security officer on an X-Road registration officer uploads a new configuration part file in the user interface.

The record is always deleted before new record with particular file name is created. The record is never modified.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| file\_name | character varying(255) |  | Name of the distributed file. Any valid file name. Cannot be NULL. |
| file\_data | bytea |  | Contents of the distributed file. Cannot be NULL. |
| content\_identifier | character varying(255) |  | Content identifier of the distributed file. The content identifier is used by security server to determine the exact type of the file. Must be unique inside an X-Road instance. Cannot be NULL. |
| file\_updated\_at | timestamp without time zone |  | Time when the distributed file was last updated. |
| ha\_node\_name | character varying(255) |  | Name of the cluster node that initiated the insertion in an HA setup; the default value in standalone setup. |

## GLOBAL\_GROUP\_MEMBERS

Join table that associates global group member identifier with the global group the member belongs to. See also documentation of the table *global\_groups*.

The record is created when a new member needs to be added to a global group. Then an X-Road registration officer adds global group member in the user interface. The record is deleted when a global group member or the group where the member belongs to is deleted in the user interface. The record is never modified.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_global\_group\_members\_on\_global\_group\_id | global\_group\_id |
| index\_global\_group\_members\_on\_group\_member\_id | group\_member\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| group\_member\_id [FK] | integer |  | ID of the member identifier that belongs to this global group. References **id** attribute of **identifiers** entity. Cannot be NULL. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |
| global\_group\_id [FK] | integer |  | ID of the global group the member referenced by *group\_member\_id* belongs to. References **id** attribute of **global\_groups** entity. Cannot be NULL. |

## GLOBAL\_GROUPS

Global group of access rights subjects that can be added to access control lists at security servers.

The record is created when a new global group needs to be added to the X-Road instance. Then an X-Road registration officer adds new global group in the user interface. The record is modified when the group description is edited or members are added to or removed from the group by an X‑Road registration officer in the user interface. The record is deleted when the global group is deleted in the user interface by an X-Road registration officer.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| group\_code | character varying(255) |  | Global group code that is unique inside the X-Road instance. Cannot be modified after the record is created. Cannot be NULL. |
| description | character varying(255) |  | Longer, human-readable description of the group. Can be modified after the record is created. |
| member\_count | integer |  | Number of members in the global group, used in user interfaces. The member count is automatically updated any time when members are added to or removed from the group. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## HISTORY

Operation (insertion, update or deletions of a record) on the tables of this database, for the purpose of auditing. Each row corresponds to the change of a single field.

The record is created in the manner described above in this document. The record can be neither modified nor deleted.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| operation | character varying(255) | NOT NULL | Name of the database operation (possible values are 'INSERT', 'UPDATE' and 'DELETE'). |
| table\_name | character varying(255) | NOT NULL | Name of the table the operation was made on. |
| record\_id | integer | NOT NULL | ID of the record that was inserted, updated or deleted, in the original table. |
| field\_name | character varying(255) | NOT NULL | Name of the column that was inserted, updated or deleted. |
| old\_value | text |  | Previous value of the column if applicable (NULL for INSERT operations). |
| new\_value | text |  | New value of the column if applicable (NULL for DELETE operations). |
| user\_name | character varying(255) | NOT NULL | Name of either the logged in user of the UI or the database user behind the connection, that initiated the operation. |
| timestamp | timestamp without time zone | NOT NULL | Date and time of the operation. |
| ha\_node\_name | character varying(255) |  | Name of the cluster node that initiated the insertion in an HA setup; the default value in standalone setup. |

## IDENTIFIERS

Identifier that can be used to identify various objects on X-Road. An identifier record is only created together with records of other entities. There is no check of duplicates when new identifier record is added. The record is deleted when any record associated with the identifier is deleted. For example, when an entity of global\_group\_members is deleted, respective identifier is deleted as well. The record is never modified.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| object\_type | character varying(255) |  | Specifies the type of the object that the identifier identifies. Possible values, defined in enum *ee.ria.xroad.common.identifier.XroadObjectType*, are 'MEMBER', 'SUBSYSTEM', 'SERVICE', 'CENTRALSERVICE', 'SERVER'. |
| xroad\_instance | character varying(255) |  | X-Road instance identifier. Present (otherwise NULL) in identifiers of all types. |
| member\_class | character varying(255) |  | Member class. Present in identifiers of 'MEMBER', 'SUBSYSTEM', 'SERVER' and 'SERVICE' type. |
| member\_code | character varying(255) |  | Member code. Present in identifiers of 'MEMBER', 'SUBSYSTEM, SERVER' and 'SERVICE' type. |
| subsystem\_code | character varying(255) |  | Subsystem code. Present in identifiers of 'SUBSYSTEM' and 'SERVICE' type. |
| service\_code | character varying(255) |  | Service code. Present in identifiers of 'SERVICE' type. |
| server\_code | character varying(255) |  | Security server code. Present in identifiers of 'SERVER' type. |
| type | character varying(255) |  | Application model class type, managed automatically by the Rails framework. Possible value are:   * 'ClientId' – if field *object\_type* has value 'MEMBER' or 'SUBSYSTEM'. * 'SecurityServerId' – if field *object\_type* has value 'SERVER'. * 'ServiceId' – if field *object\_type* has value 'SERVICE' or 'CENTRALSERVICE'. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |
| service\_version | character varying(255) |  | X-Road service version. May be present in identifiers of 'SERVICE' type. |

## MEMBER\_CLASSES

Member class supported by this X-Road instance. Member class has the purpose of grouping members with similar properties. A member class must have unique code inside the X-Road instance.

The record is added when the X-Road instance needs new member class. Then an X-Road system administrator adds a new member class in the user interface. The record is deleted when the member class is no longer necessary for this X-Road instance. Then an X-Road system administrator deletes the member class in the user interface. The description of the member class can be edited in the user interface.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| code | character varying(255) |  | Member class code, unique inside an X-Road instance. Cannot be NULL. |
| description | character varying(255) |  | Member class description. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## OCSP\_INFOS

Information about OCSP service that is offered by a particular CA. See also documentation of table *approved\_cas*.

The record is created when a new OCSP responder needs to be registered for either top CA or intermediate CA of approved CA (see also documentation of tables *approved\_cas* and *ca\_infos*). Then an X-Road system administrator adds new OCSP info in the user interface. The record can be modified or deleted in the user interface.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_ocsp\_infos\_on\_ca\_info\_id | ca\_info\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key. |
| url | character varying(255) |  | URL of the OCSP server. Must correspond to the URL format. Cannot be NULL. |
| cert | bytea |  | Certificate used by the OCSP server to sign OCSP responses (in DER encoding). |
| ca\_info\_id [FK] | integer |  | ID of the CA info record this OCSP info belongs to. References **id** attribute of **ca\_infos** entity. Cannot be NULL. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## REQUEST\_PROCESSINGS

Processing status of the management request. **Management requests** are means of managing clients and authentication certificates of security servers. See also documentation of the table *management\_requests*. Request processing binds together two management requests that refer to the same data but have different origin (security server or user interface of the central server). If one request associated with the processing is from center, the other one must be from security server and vice versa. Request processing can have one of following statuses:

* WAITING – central server has received a request that does not have the complimentary request. From this state the central server starts waiting for the complementary request.
* SUBMITTED FOR APPROVAL – central server has received two complementary requests from different sources. From this state, the user must either approve or decline the request.
* APPROVED – when the user approves the request, the processing enters APPROVED state. When entering this state the requested action (such as adding a client to a security server) is performed.
* REVOKED – when the processing is in WAITING state, respective deletion request can be sent to revoke the request. Deletion request can be sent either from security server or using the user interface.
* DECLINED – when the processing is in SUBMITTED FOR APPROVAL state, it can be declined from the user interface if X-Road registration officer decides so.

Request processing record is created when one request that can have processing (registration requests for X-Road client and security server authentication certificate) is either sent from security server or inserted in the user interface by an X-Road registration officer. Modifications to the record are related to changes of the request processing status and are described above in this section. The record is never deleted.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| type | character varying(255) |  | Application model class type, managed automatically by the Rails framework. Possible values are 'ClientRegProcessing' and 'AuthCertRegProcessing'. |
| status | character varying(255) |  | Current status of the request processing. Possible values are 'NEW', 'WAITING', 'EXECUTING', 'SUBMITTED FOR APPROVAL', 'APPROVED', 'DECLINED' and 'REVOKED'. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## REQUESTS

Management request for creating or deleting association between X-Road member and security server. Management requests are divided into registration and deletion requests.

* Registration requests are submitted to the X-Road center over two channels which bring with it the need of two complementary requests: one request is submitted through X-Road security server, the other is presented to the central server's administrator through means independent of X-Road (for example, over digitally signed e-mail). The request can be either approved or declined in the user interface of the central server. There are two types of registration requests: registration of a security server client and registration of security server's authentication certificate.
* Deletion requests are there to delete associations between X-Road clients, security servers and authentication certificates. Deletion requests are not associated with request processing. There are two types of deletion requests: deletion of security server's authentication certificate and deletion of security server' client. Deletion request can be sent for following purposes:
  + if a registration request is mistakenly sent (either from user interface of the central server or security server), respective (with the same client, security server and/or authentication certificate data) deletion request can be sent to delete the bad registration request;
  + if authentication certificate of central server needs to be deleted, respective authentication certificate deletion request is sent either from user interface of the central server or security server;
  + if client of a security server needs to be removed, respective deletion request can be sent.

The record is created in the manner described above in this section. The record is modified on following occasions:

* When the waiting request is revoked – field *processing\_status* changes. See documentations of the field *processing\_status* and the table *request\_processings* for details.
* When the request submitted for approval is either approved or declined – field *processing\_status* changes. See documentations of the field *processing\_status* and the table *request\_processings* for details.
* For a request from the X-Road center – when complementary request (from security server) is received – field *processing\_status* changes. See documentations of the field *processing\_status* and the table *request\_processings* for details.
* If a name of a member associated with the request is edited – either column *server\_owner\_name* or *server\_user\_name* changes.

The record is never deleted.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_requests\_on\_request\_processing\_id | request\_processing\_id |
| index\_requests\_on\_sec\_serv\_user\_id | sec\_serv\_user\_id |
| index\_requests\_on\_security\_server\_id | security\_server\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| request\_processing\_id [FK] | integer |  | ID of the registration processing object that manages this request. Applicable if *type* has value 'AuthCertRegRequest' or 'ClientRegRequest', otherwise NULL. References **id** attribute of **request\_processings** entity. |
| type | character varying(255) |  | Application model class type, managed automatically by the Rails framework. Possible values are 'AuthCertRegRequest', 'ClientRegRequest', 'AuthCertDeletionRequest' and 'ClientDeletionRequest'. |
| security\_server\_id [FK] | integer |  | ID of the security server related to the request.. References **id** attribute of **identifiers** entity. Cannot be NULL. |
| sec\_serv\_user\_id [FK] | integer |  | ID of the security server client related to the request. Applicable when type is client registration request or client deletion request, otherwise NULL. References **id** attribute of **identifiers** entity. |
| auth\_cert | bytea |  | Applicable for authentication certificate registration or deletion request, otherwise NULL. Contents of the authentication certificate (in DER encoding). |
| address | character varying(255) |  | Security server address for helping X-Road clients to locate the security server. Added into the global configuration when authentication certificate registration request is approved. Applicable only for requests of type 'AuthCertRegRequest', otherwise NULL. |
| origin | character varying(255) |  | Specifies where the request is from either CENTER or from SECURITY\_SERVER. |
| server\_owner\_name | character varying(255) |  | Security server owner name (for displaying in the user interface). |
| server\_user\_name | character varying(255) |  | Security server user name (for displaying in the user interface). |
| comments | text |  | Comments related to creating this request. Currently not in use. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |
| server\_owner\_class | character varying(255) |  | Security server owner class (for displaying in the user interface). |
| server\_owner\_code | character varying(255) |  | Security server owner code (for displaying in the user interface). |
| server\_code | character varying(255) |  | Security server code (for displaying in the user interface). |
| processing\_status | character varying(255) |  | Request processing status (for displaying in the user interface). Has always the same value as column status in the request\_processings table of the processing associated with this request. When there is no processing for request, the value is NULL. See also documentation of the table *request\_processings*. |

## SCHEMA\_MIGRATIONS

Keeps track of database migrations that have already been executed. Managed automatically by the Rails framework (see also <http://guides.rubyonrails.org/v3.2.21/migrations.html>).

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| unique\_schema\_migrations | version |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| version | character varying(255) | NOT NULL | UTC timestamp in 'yyyyMMddhhmmss' format describing the time when migration was executed. |

## SECURITY\_CATEGORIES

Security category, intended to be applied to security servers. **NB! Not used at the moment.**

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| code | character varying(255) |  | Security category code, must be unique. Cannot be NULL. |
| description | character varying(255) |  | Security category description. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## SECURITY\_SERVER\_CLIENT\_NAMES

Helper table that facilitates displaying security server clients in user interfaces. The table simplifies adding name to client's data.

The record is created automatically whenever new identifier of type MEMBER or SUBSYSTEM is created. See also documentation of the table *identifiers*. The record is removed automatically whenever new identifier of type MEMBER or SUBSYSTEM is removed. The record is never modified.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_security\_server\_client\_names\_on\_client\_identifier\_id | client\_identifier\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| name | character varying(255) |  | Security server client name, used in user interfaces. |
| client\_identifier\_id [FK] | integer |  | ID of the client identifier the name is associated with. References **id** attribute of **identifiers** entity. Cannot be NULL. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## SECURITY\_SERVER\_CLIENTS

Contains X-Road members or subsystems. The subject that can be associated with a security server. There are two types of associations:

* X-Road members can be security server owners. Members cannot provide and consume regular X-Road services (but can consume the X-Road management services).
* X-Road subsystems can be security server clients and can provide or consume X-Road services.

The way records are added depends on whether the record holds a member or a subsystem:

* members can be added in the user interface by the X-Road registration officer;
* subsystem can be added in the user interface by the X-Road registration officer and are also added when a client registration request for registering a subsystem that does not previously exist in this table is approved. See also documentation for tables requests and request\_processings.

The record is modified when the X-Road registration officer edits the member's name in the user interface.

The record can be deleted in the user interface by an X-Road registration officer.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_security\_server\_clients\_on\_member\_class\_id | member\_class\_id |
| index\_security\_server\_clients\_on\_server\_client\_id | server\_client\_id |
| index\_security\_server\_clients\_on\_xroad\_member\_id | xroad\_member\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| member\_code | character varying(255) |  | Member code, unique inside member class. NULL if type is 'Subsystem'. |
| subsystem\_code | character varying(255) |  | Subsystem code, unique inside member, filled if type is 'Subsystem'. NULL if type is 'XroadMember'. |
| name | character varying(255) |  | Member (human-readable) name. |
| xroad\_member\_id [FK] | integer |  | ID of the member the subsystem record belongs to. Filled if type is 'Subsystem'. NULL if type is 'XroadMember'. References **id** attribute of **security\_server\_clients** entity. |
| member\_class\_id [FK] | integer |  | ID of the the member record belongs to. Filled if type is 'XroadMember'. References ***id*** attribute of *member\_classes* entity. Cannot be NULL. |
| server\_client\_id [FK] | integer |  | Full identifier of the client. References **id** attribute of **identifiers** entity. Cannot be NULL. |
| type | character varying(255) |  | Application model class type, managed automatically by the Rails framework. Possible values are 'XroadMember' and 'Subsystem'. |
| administrative\_contact | character varying(255) |  | Administrative contact of the member, may be e-mail, phone etc. **NB! Not used at the moment!** |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## SECURITY\_SERVERS

Information about a security server registered in this X-Road instance. Security server always belongs to a particular X-Road member. For security server to function properly, it needs at least one authentication certificate. Security server may have clients (subsystems).

A prerequisite for creating the record is that a pair of authentication certificate registration requests for not yet existing security server are submitted for approval (see also documentation of tables *requests* and *request\_processings*). The record is created when one of requests submitted for approval is approved by an X-Road registration officer in the user interface. The record is modified when an X-Road registration officer edits security server address in the user interface. The record can be deleted in the user interface by an X-Road registration officer.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_security\_servers\_on\_xroad\_member\_id | xroad\_member\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| server\_code | character varying(255) |  | Security server code, unique between security servers belonging to the same owner. Cannot be NULL. |
| owner\_id [FK] | integer |  | ID of the X-Road member that owns the security server. References **id** attribute of **security\_server\_clients** entity. Cannot be NULL. |
| address | character varying(255) |  | DNS name or IP-address of the security server. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |

## SECURITY\_SERVERS\_SECURITY\_CATEGORIES

Join table enabling many-to-many relationship between security servers and security categories. In other words, associates security servers with security categories. **NB! Not used at the moment**!

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_server\_category\_to\_server\_id | security\_server\_id |
| index\_server\_to\_category | security\_category\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| security\_server\_id [FK] | integer | NOT NULL | ID of the security server with the security category referenced by *security\_category\_id*. References **id** attribute of **security\_servers** entity. |
| security\_category\_id [FK] | integer | NOT NULL | ID of the security category. References **id** attribute of **security\_categories** entity. |
| id [PK] | integer | NOT NULL | Primary key. |

## SERVER\_CLIENTS

Join table enabling many-to-many relationship between security servers and security server clients. In other words, associates security servers with its clients.

The record is created when a new client is added to the security server. It requires approval of a client registration request (see documentation of tables *requests* and *request\_processings* for details). An X-Road registration officer can do it in the user interface. The record is deleted when a client of a security server is deleted in the user interface by an X-Road registration officer. The record is never modified.

### Indexes

|  |  |
| --- | --- |
| **Name** | **Columns** |
| index\_server\_clients\_on\_security\_server\_client\_id | security\_server\_client\_id |
| index\_server\_clients\_on\_security\_server\_id | security\_server\_id |

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| security\_server\_id [FK] | integer | NOT NULL | ID of the security server. References **id** attribute of **security\_servers** entity. |
| security\_server\_client\_id [FK] | integer | NOT NULL | ID of the client the security server has. References **id** attribute of **security\_server\_clients** entity. |
| id [PK] | integer | NOT NULL | Primary key |

## SYSTEM\_PARAMETERS

System configuration parameter necessary for proper functioning of central server and entire X-Road for that matter. System parameters are stored as key-value pairs. Following is the list of supported system parameters. In an HA setup, the name of the node that initiated a particular insertion, is not significant, except for where stated explicitly.

1. **managementServiceProviderClass** – Member class part of the identifier pointing to the security server client that provides management services. The value can be changed in the user interface.
2. **managementServiceProviderCode** – Member code part of the identifier pointing to the security server client that provides management services. The value can be changed in the user interface.
3. **managementServiceProviderSubsystem** – Subsystem code part of the identifier pointing to the security server client that provides management services. The value can be changed in the user interface.
4. **centralServerAddress** –the DNS name of this central server. The value can be changed in the user interface. In an HA setup, the value is local to each node of the cluster.
5. **instanceIdentifier** – the instance identifier of this X-Road instance. Must be globally unique. The value is assigned during the initialization of the central server in the user interface.
6. **authCertRegUrl** – URL where security servers can send their authentication certificate registration requests. May contain placeholder %{centralServerAddress} which will be replaced with value of the *centralServerAddress* system parameter.
7. **confSignAlgoId** – identifier of the algorithm that is used for signing the global configuration. Supported values are 'SHA512withRSA', 'SHA256withRSA' and 'SHA384withRSA'.
8. **confHashAlgoUri** – URI of the algorithm that is used to hash distributable global configuration files. Supported values are <http://www.w3.org/2001/04/xmlenc#sha512> and <http://www.w3.org/2001/04/xmlenc#sha256>.
9. **confSignCertHashAlgoUri** – URI of the algorithm that is used to hash global configuration signing certificate. Supported values are <http://www.w3.org/2001/04/xmlenc#sha512> and <http://www.w3.org/2001/04/xmlenc#sha256>.
10. **securityServerOwnersGroup** – name of the global group where all the members that get ownership of any security server are automatically added.
11. **confExpireIntervalSeconds** – time (in seconds) during which generated global configuration is considered valid.
12. **ocspFreshnessSeconds** – time (in seconds) during which security servers should consider validity information to be usable. After that time, cached OCSP responses must be discarded. This configuration parameter is distributed to security servers as part of global configuration.

Some system parameters can be modified by an X-Road security officer in the user interface. All the system parameters that cannot be changed in the user interface, are assigned default values during the initialization of the central server. Later these can only be changed from the database. As these parameters are critical for functioning of entire X-Road instance, these must be modified with extreme care.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| key | character varying(255) |  | System parameter key. Cannot be NULL. |
| value | character varying(255) |  | System parameter value corresponding to the key. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |
| ha\_node\_name | character varying(255) |  | Name of the cluster node that initiated the insertion in an HA setup; the default value in standalone setup. |

## TRUSTED\_ANCHORS

Trusted anchor of a federation partner. A trusted anchor is the configuration anchor of the configuration source distributing the external configuration of a federation partner.

The record is created or modified exactly the same way as described in the documentation of table *anchor\_url\_certs.* The record is never modified.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| instance\_identifier | character varying(255) |  | Instance identifier of the trusted anchor. Cannot be NULL. |
| trusted\_anchor\_file | bytea |  | Trusted anchor file (in XML format). Cannot be NULL. |
| trusted\_anchor\_hash | text |  | Hash of the trusted anchor file. Cannot be NULL. |
| created\_at | timestamp without time zone |  | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone |  | Record last modified time, managed automatically by the Rails framework. |
| generated\_at | timestamp without time zone |  | Anchor generation time (read from the anchor file). |

## UI\_USERS

UI user name with its last used locale. Maps possible user interface (UI) user names with locales so that when UI user is logged in next time, the locale it has been used is remembered. If a user with no assigned locale logs in, the first available locale is selected to this user. Later user can change its locale in the user interface.

The record is created when the user is logged in the user interface for the first time. The record is modified when the user logged in changes its locale in the user interface. The record is never deleted.

### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Modifiers** | **Description** |
| id [PK] | integer | NOT NULL | Primary key |
| username | character varying(255) |  | User name of the UI user. |
| locale | character varying(255) |  | Current locale of the UI user. |
| created\_at | timestamp without time zone | NOT NULL | Record creation time, managed automatically by the Rails framework. |
| updated\_at | timestamp without time zone | NOT NULL | Record last modified time, managed automatically by the Rails framework. |