|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| Controls Configuration Database (CCDB)  Design Document |
|  |
|  |

**APROVAL MATRIX**

| Role | Name | Affiliation | Date |
| --- | --- | --- | --- |
| Author | Miha Vitorovič | Cosylab | 2014-07-20 |
| Reviewer | Suzanne Gysin | ESS AB |  |
|  | Timo Korhonen | ESS AB |  |
|  | Daniel Piso Fernández | ESS AB |  |
|  | Miha Reščič | ESS AB |  |
|  | Ricardo Fernandes | ESS ERIC | 2016-04-07 |
| Approver | Suzanne Gysin | ESS AB |  |
|  | Garry Trahern | ESS AB |  |
|  | Ricardo Fernandes | ESS ERIC | 2016-04-07 |

**VERSION SUMMARY**

| Version | Date | Modifications since last revision |
| --- | --- | --- |
| 1.0 | 2014-05-03 | Initial document |
| 2.0 | 2014-07-17 | Merging with the DISCS model |
| 2.1 | 2015-10-08 | Updated information on the RESTful service to mirror the actual implementation |
| 2.2 | 2016-03-08 | Updated design according to actual implementation |
| 2.3 | 2016-07-20 | Rewritten section 3.4.1 to correctly describe the RESTful interface. |

Summary

The CCDB is an application to hold ESS controls configuration data; in particular static data required to install and maintain the equipment in the scope of the ICS service agreements. It will contain information on the devices that play a role in the Integrated control system, which means at least all devices that will have EPICS signals, devices that control them, and also devices and other entities that are conceptually important to the application users.

Table of Contents

Summary 3

List of Tables 6

List of Figures 7

List of Abbreviations 9

1. Introduction 10

2. Architecture 10

3. Design 12

3.1 Database 12

3.1.1 Change History 12

3.1.2 Data Definitions 13

3.1.3 Device Instances 14

3.1.4 Relationship 14

3.1.5 Device Types 14

3.1.6 Data Definition Details 14

3.1.6.1 Units 15

3.1.6.2 Enumerations 15

3.1.6.3 Properties, Tags and Property Values 16

3.1.6.4 Containers and Installation Slots 17

3.1.6.5 Device Types 19

3.1.7 User Data 20

3.1.7.1 Device Instance 20

3.1.7.2 Artifacts 21

3.1.7.3 Property Values 21

3.1.7.4 Tags 21

3.1.8 History Control 22

3.1.8.1 Audited Entities 23

3.1.8.2 Audit Trail Description 23

3.1.8.3 Examples 23

3.2 User Roles and Permissions 30

3.2.1 Configuration Module Authorization Model 31

3.2.2 Entity Based Authorization 31

3.2.2.1 Device Types 32

3.2.2.2 Devices 32

3.2.2.3 Containers and Installation Slots 32

3.3 UI Layer 33

3.3.1 Data Definitions 33

3.3.1.1 Units 33

3.3.1.2 Enumerations 35

3.3.1.3 Properties 37

3.3.1.4 Device Type 39

3.3.1.5 Device Type Properties 41

3.3.1.6 Containers 43

3.3.1.7 Installation Slots 45

3.3.1.8 Device Installation 47

3.3.2 Adding and Browsing Device Instances 48

3.3.2.1 Device Types 48

3.3.2.2 Adding Device Instance 49

3.3.2.3 Device Instance Details 50

3.3.3 Importing Data 51

3.3.3.1 Data Import 51

3.3.3.2 Excel File 51

3.3.4 Generating Reports 52

3.3.5 Data Export 54

3.3.6 Deleted Entries 54

3.4 Service Layer 55

3.4.1 RESTful Interface 55

3.4.1.1 Device Types 55

3.4.1.2 Installation Slots 57

3.4.1.3 Installation Slot Names 61

3.4.1.4 Device Instances 62

4. Requirements Traceability 65

5. References 66

List of Tables

[Table 1 Possible “contains” relationship endpoints 18](#_Toc456854665)

[Table 2 History record description 22](#_Toc456854666)

[Table 3 Enumeration creation entry 24](#_Toc456854667)

[Table 4 Enumeration modification entry 24](#_Toc456854668)

[Table 5 Enumeration deletion entry 25](#_Toc456854669)

[Table 6 Unit creation entry 25](#_Toc456854670)

[Table 7 Property creation entry 25](#_Toc456854671)

[Table 8 Device type creation entry 26](#_Toc456854672)

[Table 9 Container creation entry 26](#_Toc456854673)

[Table 10 Adding relationships 27](#_Toc456854674)

[Table 11 Installation slot creation entry 27](#_Toc456854675)

[Table 12 Device type add property definition 28](#_Toc456854676)

[Table 13 Add device type property 28](#_Toc456854677)

[Table 14 Device type remove property definition 29](#_Toc456854678)

[Table 15 Device type adding and removing tags 30](#_Toc456854679)

[Table 16 Device property types 50](#_Toc456854680)

[Table 17 Requirements traceability 65](#_Toc456854681)

List of Figures

[Figure 1 Three tier architecture 10](#_Toc456854682)

[Figure 2 Java EE architecture 11](#_Toc456854683)

[Figure 3 CCDB integration with other ICS services 12](#_Toc456854684)

[Figure 4 CCDB application object model (CCDB\_030) 13](#_Toc456854685)

[Figure 5 OM detail: Data type, Unit and Property 15](#_Toc456854686)

[Figure 6 OM detail: Slot entity (containers and installation slots) (CCDB\_050) 17](#_Toc456854687)

[Figure 7 OM detail: device types 19](#_Toc456854688)

[Figure 8 OM detail: device instances 20](#_Toc456854689)

[Figure 9 OM detail: Property values and tags 21](#_Toc456854690)

[Figure 10 Unit definitions 34](#_Toc456854691)

[Figure 11 Unit definition 35](#_Toc456854692)

[Figure 12 Browse enumerations screen 36](#_Toc456854693)

[Figure 13 Defining enumeration 37](#_Toc456854694)

[Figure 14 Browse properties screen 38](#_Toc456854695)

[Figure 15 Property definition screen 39](#_Toc456854696)

[Figure 16 Device type listing 40](#_Toc456854697)

[Figure 17 Define device type 41](#_Toc456854698)

[Figure 18 Defining device type properties 41](#_Toc456854699)

[Figure 19 Adding device type property 42](#_Toc456854700)

[Figure 20 Adding or editing a tag 43](#_Toc456854701)

[Figure 21 Browse containers 44](#_Toc456854702)

[Figure 22 Define container 44](#_Toc456854703)

[Figure 23 Browse installation slots 45](#_Toc456854704)

[Figure 24 New installation slot 46](#_Toc456854705)

[Figure 25 Installation slot properties 46](#_Toc456854706)

[Figure 26 Adding relationships to the installation slots 47](#_Toc456854707)

[Figure 27 Device installation 48](#_Toc456854708)

[Figure 28 Listing device instances 49](#_Toc456854709)

[Figure 29 Adding a new physical device (CCDB\_060, CCDB\_070, CCDB\_073, CCDB\_110) 49](#_Toc456854710)

[Figure 30 Data import dialog 51](#_Toc456854711)

[Figure 31 Report generation screen 53](#_Toc456854712)

[Figure 32 Audit log entries 54](#_Toc456854713)

[Figure 33 Audit log entry details 55](#_Toc456854714)

[Figure 34 Sample RESTful service output listing all device type 56](#_Toc456854715)

[Figure 35 Sample RESTful service output showing a specific device type 57](#_Toc456854716)

[Figure 36 Sample RESTful service output listing all installation slots 58](#_Toc456854717)

[Figure 37 Sample RESTful service output listing all installation slots of specified device type 59](#_Toc456854718)

[Figure 38 Sample RESTful service output showing a specific installation slot 60](#_Toc456854719)

[Figure 39 Sample RESTful service output listing all installation slot and container names 61](#_Toc456854720)

[Figure 40 Sample RESTful service output listing installation slots of a specified device type 62](#_Toc456854721)

[Figure 41 Sample RESTful service output listing all devices 63](#_Toc456854722)

[Figure 42 Sample RESTful service output showing specific device 64](#_Toc456854723)

List of Abbreviations

| Abbreviation | Definition |
| --- | --- |
| CCDB | Controls Configuration Database |
| CCDB\_xxx | Reference to a CCDB requirement (see 4. Requirements Traceability) |
| CSV | Comma-separated Values |
| DISCS | Distributed Information Services for Control Systems |
| ICS | Integrated Control System |
| Java EE | Java Enterprise Edition |
| JPA | Java Persistence API |
| JSF | JavaServer Faces |
| ORM | Object-Relational Mapping |
| OM | Object model |
| POJO | Plain Old Java Object |
| RBAC | Role Based Access Control |
| REST | Representational State Transfer |
| RF | Radio Frequency |
| SEDS | Serialization of EPICS Datatype Standards |
| UI | User Interface |
| URL | Uniform Resource Locator |

# Introduction

The Controls Configuration Database (CCDB) is the main database to hold information on the equipment in the scope of the ICS service agreements. Besides being a catalog of the equipment, it holds static configuration information for those devices, like:

* Type
* Nominal parameters
* Physical attributes (size)
* Item related information (manufacturer, model, S/N, etc.)
* Physical location

At this time this includes equipment related to Machine Protection, Personnel Protection, timing, vacuum, cryogenics, RF and conventional facilities.

Besides being storage for such information, the CCDB also provides a Web UI for browsing, adding and modifying such information and generation of various reports. The application also provides a service layer which other applications can use to query the contents of the CCDB.

The CCDB application is being developed as the configuration module in the DISCS [3] collaboration.

# Architecture

The CCDB will be implemented in a classical three tier architecture consisting of the data storage, business logic and presentation layers. The application will use Java EE technology and will implement a user interface web client and a RESTful service for other types of clients.

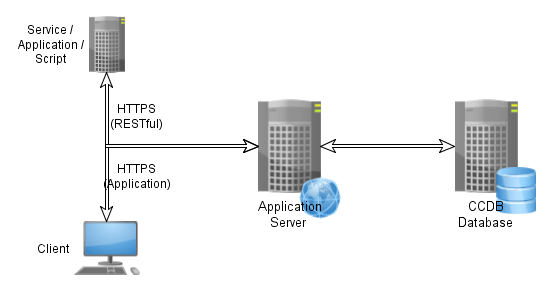


Figure 1 Three tier architecture

The database will store Control System Configuration information in a relational database. The database will contain information required to install and maintain the equipment related to timing, vacuum, cryogenics, RF, machine protection, personnel protection, etc…

The application will access and present the information using the Java EE (CCDB\_195) application server platform and technologies. The business logic will access the data through a JPA 2.1 database abstraction layer. CCDB will expose this data through UI client to the end users, or through a RESTful service to other services, applications and any additional UI clients that may be written.

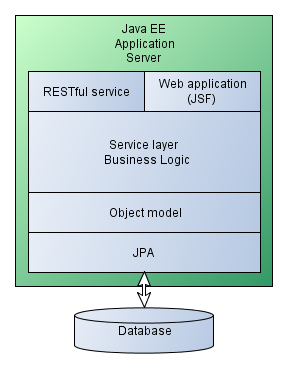


Figure 2 Java EE architecture

Other ICS services will handle parts of the information about ICS and the CCDB will need to integrate with them (CCDB\_175) to provide a complete service to the ICS users. These services include:

* Cabling Database
* Naming service
* Device Configuration Database
* RBAC

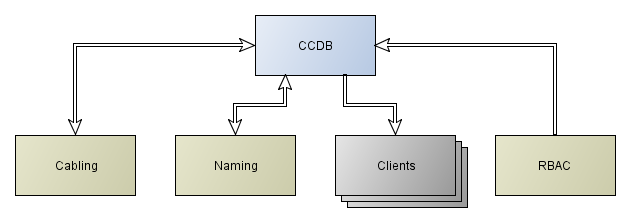


Figure 3 CCDB integration with other ICS services

# Design

As seen in the architecture, the application is composed of a database holding all the data, a business logic layer, an application front end for editing the data and a RESTful service for accessing the data from other ICS related services.

## Database

The application will use relational database as its back-end storage for entities. An application entity represents an object stored in the database; in database terminology it is a line in a database table.

### Change History

The database will support entity auditing, meaning that for each entity it will be possible to retrieve the history of property changes. The history will be stored into the database and will record basic information:

* Entity creation/modification/deletion
* The timestamp of the action
* The user performing the action

Through this the users will be able to see the entire history of the CCDB which can be used for auditing, and any CCDB entity can be reconstructed using this history.

One thing that the CCDB will not support is data branching, meaning that the users will not be able to create parallel and independent version of data set.

The section describes the CCDB object model and the database schema is constructed from that model automatically by the JPA layer. The document describes an object model based on the database used by the DISCS collaboration module. Figure 4 shows a schematic view of the object model with basic relationships between the entities. The model is explained in greater detail later on.



Figure 4 CCDB application object model (CCDB\_030)

The data in the CCDB can be divided into three categories:

1. Data definitions
2. Device instances
3. Relationships

### Data Definitions

The data definitions are used to define the data that will be contained in the database, and include:

* Enumeration
* Unit definitions
* Property definitions
* Containers and installation slots (see 3.1.6.4)
* Device types

The unit definitions and enumerations are used for defining properties, and property instances can be associated with containers, device types and device instances. Only properties that are defined in the CCDB can be used in this way. Property definitions can only be added by the CCDB super user and ordinary users can only use properties that are already defined.

### Device Instances

Device instances are physical objects used in the ICS. The device instances can only be of the types defined in the CCDB database.

### Relationship

Relationships define different relationships between database entities (CCDB\_150). The database supports arbitrary number of relationships between entities, but the relationships are tightly coupled with the business logic, and their definition is part of the application design. CCDB will support the following types of relationships:

* Contains
* Controls
* Powers

The contains relationship defines the basic hierarchy in the database. It defines relationships between containers themselves and between the containers and installation slots.

The controls and powers relationships can only be defined between installation slots. The relationships of these types are used for displaying links between different devices.

### Device Types

The device type specifies the details of the type of similar devices (CCDB\_010, CCDB\_020, CCDB\_025, CCDB\_027, CCDB\_028). The device type specifies the name of the device type, and can have various properties associated. Each specific type of devices will have a separate device type entity.

The device type specifies the properties that are common for all device instances, like *instruction manual*, *subject matter expert*, *technical specifications*, etc.

The device type defines the set of properties for its device instances.

### Data Definition Details

Before storing the devices into the CCDB, the super user will have to specify the definitions that will be used by the users of the CCDB.

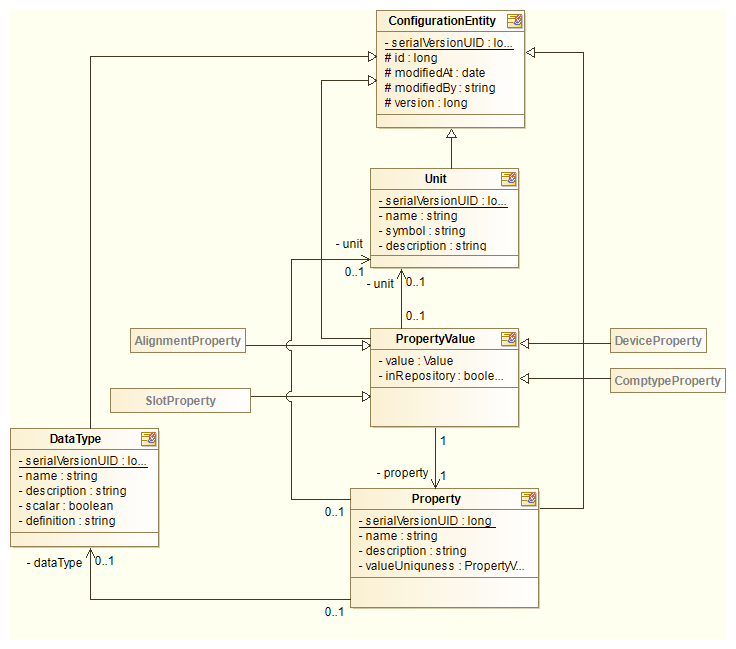


Figure 5 OM detail: Data type, Unit and Property

#### Units

Units describe the physical units that can be used with different database properties. The goal of defining units is standardizing the input values and minimizing the confusion of the users. Each unit has a symbol and a description. For example:

* Name: Ampere
* Symbol: A
* Description: Base SI unit for electric current

When deleting a unit the CCDB application has to check if this unit is used in any property definition. If yes, deletion has to fail.

#### Enumerations

For property values it is possible to use any of the user defined enumerations. An enumeration is a predefined set of values and the user can only choose one of the predefined values (CCDB\_110, CCDB\_111). For implementing enumerations the DISCS project SEDS enumeration type will be used.

Each enumeration has the following fields:

* Name
* Description
* Definition

The values are defined in multiple lines, each line containing a name of the value option. Example:

* In fabrication
* Under testing
* Under repair
* Ready
* Spare

The value option text can contain only alphanumeric characters, space, underscore and a dash character. No other punctuation or special characters can be used.

The enumeration definition is stored as a new data type. The possible values of the enumeration are stored into the definition field as a SEDS enumeration in JSON encoding. Enumerations are the only data type entities that can be deleted from the database.

When deleting an enumeration the CCDB application has to check if this enumeration is used in any property definition. If yes, deletion has to fail.

#### Properties, Tags and Property Values

Properties are different values (CCDB\_170, CCDB\_080, CCDB\_090, CCDB\_106, CCDB\_107, CCDB\_110) that can be assigned to various database entities, like container, installation slot, device type and device instance. Each property definition has:

* Name
* Description
* Data type
* Unit (optional)
* Uniqueness (see below)

The data type of the property will be one of the following basic data types:

* Integer number
* Double precision floating point
* String of characters (text)
* Date and time
* URL (string of characters which is known to contain URL)
* Vector of integer numbers (1D array)
* Vector of double precision numbers (1D array)
* List of strings (1D array)
* Table of double precision numbers (2D array)
* User defined enumerations

For storing tables (2D arrays) the application will use SEDS table data type, which will be stored in the database in a serialized form.

The property can be associated with following types of entities in the database:

* Device type
* Container
* Installation slot
* Device instance
* Alignment record

Each property value is stored in the database as a SEDS value in the serialized form.

The uniqueness field specifies how the business layer checks whether the property value is allowed. The possible values for uniqueness field are:

* None (the value of the property is not checked)
* Device type (no installation slots or devices of the same data type can have the same value of this property)
* Universal (no other entity in the database can have the same value of this property)

These checks are performed when the user tries to set the value of the property.

When deleting the property definition the CCDB application must check whether the property definition is used in any property value or in any device type as a device type property definition. If yes, deletion has to fail.

#### Containers and Installation Slots

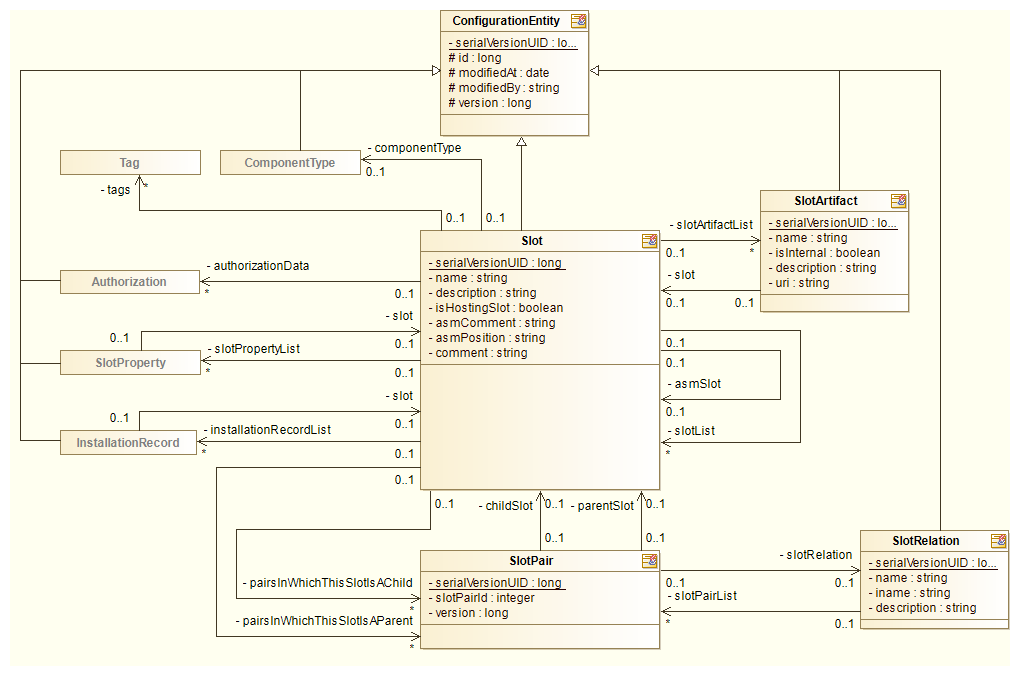


Figure 6 OM detail: Slot entity (containers and installation slots) (CCDB\_050)

Containers are database entities that contain other containers or installation slots. The containers are purely user defined and can be either logical or physical. Examples are:

* System
* Subsystem
* Section
* Group
* Building
* Room
* Rack

The containers can be used as a parent or child in the *contains* relationship (CCDB\_150). The *contains* relationship will have two endpoints; the parent and child of a relationship. The relationship will not be strictly free, possible endpoints of the relationship will be dictated by the business logic and are listed in the table below.

The difference between a *container* and an *installation slot* is that for *installation slot* the user can select the device type to be installed into it. The installation slot can also have properties like its beamline position and its global coordinates.

Each container implicitly has a component type of \_GRP or \_ROOT. The \_ROOT component type is assigned to all containers that do not participate in the in the *contains* relationship as a child. All other containers have a component type of \_GRP.

| Table 1 Possible “contains” relationship endpoints | |
| --- | --- |
| Parent | Child |
| Container | Container |
| Container | Installation slot |
| Installation slot | Installation slot |

When deleting a container the CCDB application has to check if this container is used in any *contains* relationship as a parent. If yes, the user is presented with the dialog listing all slots that will be delete as a result and user has to confirm deletion. All relationships in which the container has a role of the child have to be removed at the same time.

The authorized users will create the installation slots through the UI. On the topmost level of the hierarchy the authorized user can only create a container. An installation slots can also have properties associated with it, but more importantly, the installation slot actually bears the naming convention name (CCDB\_040, CCDB\_050) (if required by the device). The naming convention name actually describes the place and the device role. A typical situation where this becomes evident is device failure. The replacement device needs to have the same name and properties as the broken device to play a role in the control system.

The installation slots are also the endpoints in the *powers* and *controls* relationship.

The installation slot also has device type. In this case it defines the type of the device instance that can be installed into it.

#### Device Types

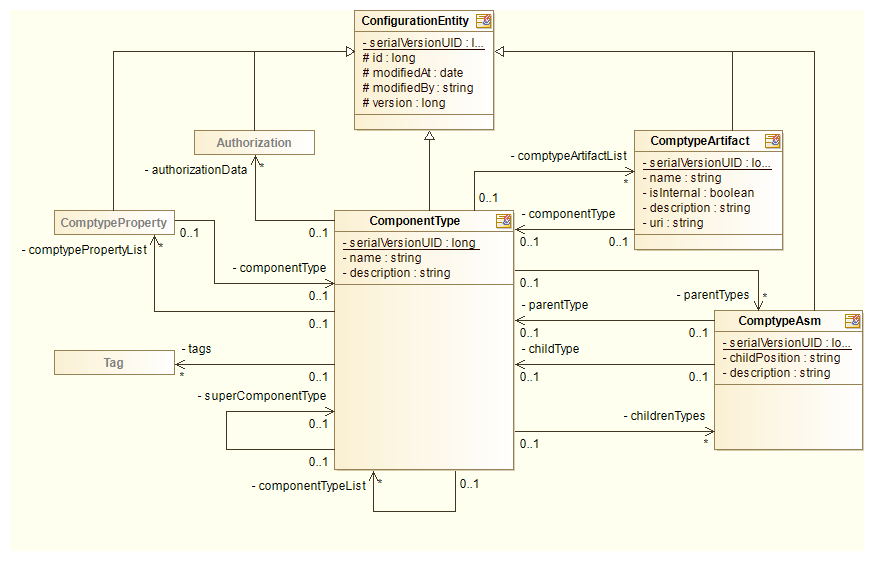


Figure 7 OM detail: device types

The device type defines the type of the device that can be used in the CCDB application. The database will only contain device instances for which the device type is defined in the database. Only the CCDB super user will be able to define or change the device types.

The device type will have a name, a description and various device type properties and device type property definitions associated with it.

The CCDB will contain a list of specific device types that are needed for its own purposes. The naming tool is foreseen to hold the master list of the specific device types in some future version. Once the naming tool gets this functionality and the interface is designed, the CCDB specific device type list will be synchronized with and linked to the master list.

The predefined fields will be the same for all device types while the properties can be any properties defined in the CCDB. Each device type will be able to have two types of properties associated with it.

Device type properties are properties that have the same value for each device instance, these are the properties of all the devices of this type. For example, the device documentation is such a property.

Device property definitions are device properties that can have different values for each device instance or installation slot, but it is known that you have to have them defined for each instance of this device type. Additional checks can make sure that all device instances and installation slots have a value set for each such property. When defining a property definition the authorized user can define a default value for such a property value. When a new device or installation slot instance is created, its property value is filled with this default.

When deleting a device type the CCDB application has to check if it is used by any device instance. If yes, deletion has to fail.

### User Data

This section describes the data accessible to authorized users.

#### Device Instance

The most common type of user data is the information on the device instances.

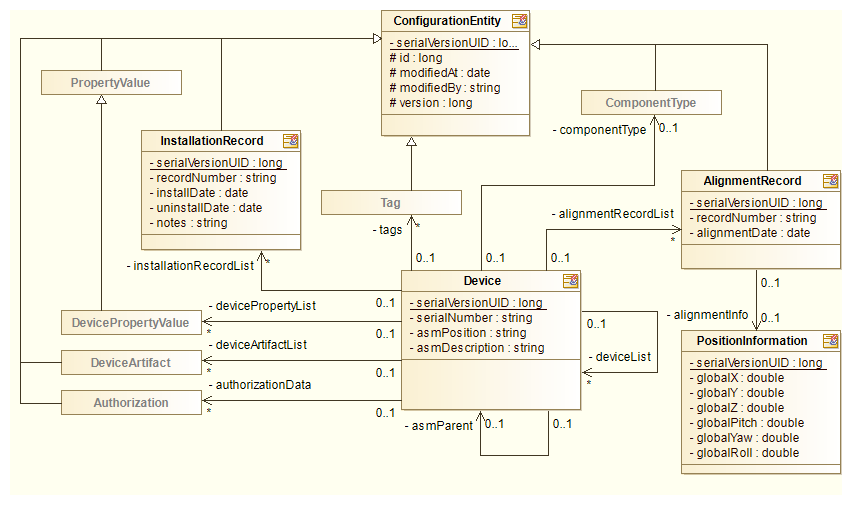


Figure 8 OM detail: device instances

A device instance holds information about a specific instance of a device. The device instance can either be installed or not. A device instance is installed if an installation record ties it to an installation slot. Only one device can be installed into each slot, and application has to check this before installing a device.

Each device can also have a number of properties (CCDB\_107) and tags associated with it.

When removing the device from the database, the device must not be installed into any slots – there must not be any installation record associated with it. If the device is installed it needs to be uninstalled first. When device is removed, all the device properties and artifacts are removed as well.

#### Artifacts

Artifacts offer a way to the user to associate some additional information to a device instance. This information can only be an URL pointing to some external resource or file attachment of some sort.

The main difference between an artifact and a property is that artifacts do not need to be defined by the application super-user (the user can create them on the fly), and there is no limit to the number of artifacts that can be associated with the database entity. In contrast, each entity can only have one property of a certain name associated with it. The same entity can have any number of artifacts named *Instructions* associated with it.

The authorized users can create artifacts for *device types,* *containers, installation slots,* and *device instances*.

**Note:** The attachment are saved on the file system into a folder specified at application installation. The system administrator should be notified to include this folder into the application backup.

#### Property Values

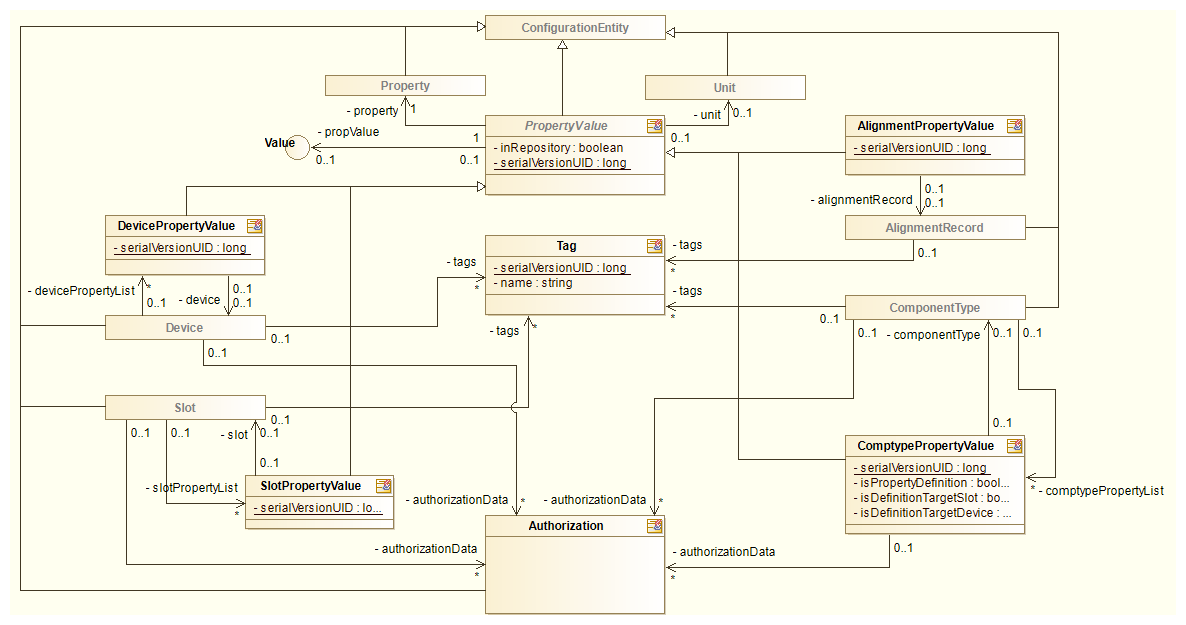


Figure 9 OM detail: Property values and tags

Property values are name value pairs associated with the database entity. The user can set the property values only for properties that were defined for a device type by the super user. The user can only set property values on the entities for which he is authorized to (see 3.2). (CCDB\_080, CCDB\_090, CCDB\_106, CCDB\_107, CCDB\_110, CCDB\_111)

#### Tags

Tags are user defined strings that can be attached to *device types*, *containers*, *device instances* and *alignment records*. Their primary purpose is marking the database entities with user defined values which can later be used when generating reports, i.e. a list of all device instances which are tagged *Beam instrumentation*.

The tags are shared among all the entities in the database, so once a tag is created and associated with an entity, it cannot be simply renamed or deleted from the database – in case of renaming, the tag would change for all associated entities, which may not be what the user wants. The user can always disassociate a tag with the entity, and create a new tag in event of an error.

### History Control

For each entity in the CCDB a history of changes will be tracked (CCDB\_140). This will be a log of all the changes that have been made by some action, and such history will be associated with each entity.

1-D and 2-D double arrays can contain very large amounts of data, so changed data of these two types the will only be logged entirely if it contains less than 100 rows and less than 50 columns, otherwise the data will simply be truncated.

1-D string list or integer array change will only be logged if it contains less than 100 elements, otherwise the audit log entry data will be truncated.

Both these values can be changed through system properties org.openepics.discs.conf. props.auditLogRows and org.openepics.discs.conf.props.auditLogColumns.

The history record will contain:

| Table 2 History record description | |
| --- | --- |
| Field | Description |
| Entity ID | The database entity this audit trail history belongs to. This field contains the unique database identifier of this entity. |
| Operation | Possible values are:   * UPDATE * CREATE * DELETE |
| User | Authenticated user performing the action. (CCDB\_140) |
| Timestamp | The date and time of the action. (CCDB\_140) |
| Entity type | The type of the entity this action was performed on.  Explanation: All actions are performed in the context of some entity, i.e. the history is logically associated with them. This context entity is referenced in the *Parent entity* field. However, the action itself may be performed on another type of record. This is noted in the *Entity type* field. For now the only examples of such action are the modification of a relationship between two entities, or an action on some property (name/value pair).  In effect the field specifies the table this action is working on. |
| Entity name | If the entity contains some user defined ID (name), it is stored in this field. |
| Change | A structured description of the values in the entity. In case the action caused some of the entity fields or properties to change (CCDB\_140), this field contains a generated report of all the values. |

Such history will provide an audit trail for each entity. User will be able to use this audit trail to inspect the changes and revert the unwanted changes manually. The audit trail history for each entity is ordered chronologically. The history consists of at least one *create* record. The subsequent data entries indicate modification/addition/deletion of one or more entity attributes. If the entity is deleted, then the last record in the history is the *delete* action. No further history records may follow this record.

To implement the audit trail logging the entity life-cycle callbacks mechanism of the Java EE framework will be used.

#### Audited Entities

The database contains the following types of entities:

* Enumeration definitions
* Units
* Container entities
* Installation slots
* Device type entities
* Physical devices
* Relationships

The database will provide a uniform history interface for all entities except *relationship* entities. The relationship is always between two installation slots and can only be added or removed. The relationship itself cannot change, so there is no audit trail log needed for it. Because of this, the audit trail history for relationships will always be recorded in the context of the entities that participate in the relationship. A separate relationship audit trail record will be created for both entities participating in the relationship.

#### Audit Trail Description

This section describes how the audit trail history is composed, and how audit trail records are mapped to the entities for which they are created.

The table lists and describes the possible *modify* entries in the audit trail history.

Relationship audit trail records are always created for both entities in the relationship.

#### Examples

This section gives examples of how the audit trails for various parent entity types are stored in the database. The log of changes will be shown to the user in the UI, as shown in Figure 32 and Figure 33.

##### Enumeration

| Table 3 Enumeration creation entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Device status” data type DB entity |
| Action | CREATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 12:01:00 |
| Entity type | Enumeration |
| Name | Device status |
| Change | {  "description" : "Describes a status of the physical device",  "scalar" : true,  "definition" : "{\"meta\":{\"type\":\"SedsEnum\",\"protocol\":\"SEDSv1\",\"version\":\"1.0.0\"},\"data\":{\"selected\":\"Installed\"},\"type\":{\"elements\":[\"INSTALLED\",\"IN\_SERVICE\",\"ON\_STOCK\",\"TESTED\"]}}"  } |

| Table 4 Enumeration modification entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Device status” enumeration DB entity |
| Action | UPDATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 12:02:11 |
| Entity type | Enumeration |
| Name | Device status |
| Change | {  "description" : "Describes a status of the physical device",  "scalar" : true,  "definition" : "{\"meta\":{\"type\":\"SedsEnum\",\"protocol\":\"SEDSv1\",\"version\":\"1.0.0\"},\"data\":{\"selected\":\"INSTALLED\"},\"type\":{\"elements\":[\"INSTALLED\",\"IN\_SERVICE\",\"ON\_STOCK\",\"TESTED\",\"LAB\_EQUIPMENT\"]}}"  } |

| Table 5 Enumeration deletion entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Device status” enumeration DB entity |
| Action | DELETE |
| User | mvitorovic |
| Timestamp | 2014-05-06 08:57:36 |
| Entity type | Enumeration |
| Name | Device status |
| Change | {  "description" : "Description",  "scalar" : true,  "definition" : "{\"meta\":{\"type\":\"SedsEnum\",\"protocol\":\"SEDSv1\",\"version\":\"1.0.0\"},\"data\":{\"selected\":\"INSTALLED\"},\"type\":{\"elements\":[\"INSTALLED\",\"IN\_SERVICE\",\"ON\_STOCK\",\"TESTED\",\"LAB\_EQUIPMENT\"]}}"  } |

##### Unit

| Table 6 Unit creation entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Ampere” unit DB entity |
| Action | CREATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 12:10:04 |
| Entity type | Unit |
| Name | Ampere |
| Change | {  "symbol" : "A",  "description" : "Electric current"  } |

##### Property

| Table 7 Property creation entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Current” property DB entity |
| Action | CREATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 12:19:54 |
| Entity type | Property |
| Name | Current |
| Change | {  "description" : "Electric current",  "valueUniqueness" : "NONE",  "dataType" : "float",  "unit" : "Ampere",  } |

##### Device type

| Table 8 Device type creation entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power supply” device type DB entity |
| Action | CREATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 14:29:17 |
| Entity type | Device type |
| Name | ControlLogix AC Power Supply |
| Change | {  "description" : "Rockwell"  } |

| Table 9 Container creation entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power equipment” device type DB entity |
| Action | CREATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 14:29:17 |
| Entity type | Container |
| Name | Power equipment |
| Change | {  "description" : "Power equipment logical group",  " hostingSlot" : false,  "componentType" : "\_GRP",  "parentSlots" : [ {  "Inventory" : "[Contained in]"  } ]  } |

| Table 10 Adding relationships | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power equipment” container DB entity |
| Action | UPDATE |
| User | Mvitorovic |
| Timestamp | 2014-05-05 14:29:17 |
| Entity type | Container |
| Name | Power Equipment |
| Change | {  "description" : "Power equipment logical group",  "hostingSlot" : false,  "componentType" : "\_GRP",  "childrenSlots" : [ {  "ISrc-01:PS-A01" : "[Contains]"  } ],  "parentSlots" : [ {  "Inventory" : "[Contained in]"  } ]  } |

| Table 11 Installation slot creation entry | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “ISrc-01:Chop-A01” device type DB entity |
| Action | CREATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 14:29:17 |
| Entity type | Installation slot |
| Name | ISrc-01:Chop-A01 |
| Change | {  "description" : "Ion source chopper",  "hostingSlot" : true,  "componentType" : "Chopper",  "parentSlots" : [ {  "ISrc" : "[Contained in]"  } ]  } |

| Table 12 Device type add property definition | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power supply” device type DB entity |
| Action | UPDATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 14:33:12 |
| Entity type | Device type |
| Name | ControlLogix AC Power Supply |
| Change | {  "description" : "Rockwell",  "devicePropertyList" : [ {  "Current" : null  }  } |

| Table 13 Add device type property | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power supply” device type DB entity |
| Action | UPDATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 14:33:48 |
| Entity type | Device type |
| Name | ControlLogix AC Power Supply |
| Change | {  "description" : "Rockwell",  "comptypePropertyList" : [ {  "Weight" : 132.42  } ],  "devicePropertyList" : [ {  "Current" : null  }  } |

| Table 14 Device type remove property definition | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power supply” device type DB entity |
| Action | UPDATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 14:33:48 |
| Entity type | Device type |
| Name | ControlLogix AC Power Supply |
| Change | {  "description" : "Rockwell",  "comptypePropertyList" : [ {  "Weight" : 132.42  } ]  } |

##### Relationship Removal

The relationship removal (usually used for removing the additional relationship from the installation slot) is recorded in the following way. As can be seen in Table 10 each container lists all its parent and child relationships in two lists, which are the snapshot of the current parent and child relationships. The removal of the relationship is simply recorded as a new snapshot. To reconstruct which relationship was deleted, this entry must be compared to the previous entry.

##### Working with tags

| Table 15 Device type adding and removing tags | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power supply” device type DB entity |
| Action | UPDATE |
| User | mvitorovic |
| Timestamp | 2014-05-05 14:33:48 |
| Entity type | Device type |
| Name | ControlLogix AC Power Supply |
| Change | {  "fields" : {  "name" : "ControlLogix AC Power Supply",  "description" : "Rockwell"  },  "properties" : null,  "tags" : {  "add" : [ "Power Supply" ],  "delete" : [ "Testing" ]  },  "relationships" : null  } |

## User Roles and Permissions

The assignment of users to roles will be done through the RBAC service and is out of the scope of the CCDB application (CCDB\_175).

The CCDB is registered in RBAC as a resource with the name CCDB.

There will be 3 types of user roles when accessing this application:

* Administrator will be able to specify data definitions and also define device instances
* Authorized users will be able to add, modify and delete device instances, and install them into the installation slots
* Non-authorized users will have read only access to the device instances

By being the only one that is able to create data definitions in the CCDB the implicit role of the administrator is also to guard CCDB against proliferation of various data definitions: statuses, units, etc.

The users with the administrator role will be the only ones able to perform tasks described in sections 3.3.1 and 3.3.3.1.

The authorized user will be able to perform the other actions described in sections 3.3.2 through 3.3.6 (except 3.3.3.1). Authorized users will also be able to grant create, modify and delete permissions to other authenticated users (see 3.2.2).

Anonymous users will be able to view the same data as authorized users, but will not be able to add, modify or delete any data.

Currently there are no requirements for finer grained permissions in the CCDB applications.

### Configuration Module Authorization Model

RBAC [2] defines authorization in the following way:

* As user authenticates with the RABC server he is assigned a collection of roles
* The roles are a collection of permissions for various resources
* The resource in this case is the CCDB application
* The permissions are defined by the application developer
* The application queries the RBAC server whether the authenticated user has a certain permission

The DISCS configuration module authorization is based on permissions on the database entities and UI elements. For each application resource the user can have the following permissions:

* Create
* Update
* Delete
* Rename

In the RBAC based authorization model the following permissions are defined:

1. WriteUnits
2. WriteEnumerations
3. WriteProperties
4. WriteDeviceTypes
5. WriteDevices
6. WriteContainersSlots

Each of the permissions allows the user granted the permission to create, modify and delete the entity he is granted permission for. The super user role has all of the above permissions. Additional roles can be created granting just some of the permissions to the user. Every authenticated user by default does not have any of the permissions, meaning he can only read and export the data CCDB.

### Entity Based Authorization

In addition to the default RBAC permissions described in the section 3.2.1, each authenticated user can also be granted access to only some of the

* Device types,
* Device type properties,
* Devices,
* Containers or Slots.

To grant special permissions to the user his user name or any RBAC defined permission for CCDB can be used. For RBAC permissions this means that in addition to the permissions listed in 3.2.1 new permissions can be defined and then used in the entity based authorization. In the UI the usernames and RBAC permissions will be represented in the following way. The username will be represented exactly as it is defined in the RBAC (LDAP), while the permission will be written inside square brackets []. For example:

* johndoe (user name)
* [MagnetsDepartment] (RBAC permission)

The entity based authorization can be used for the entities listed above in the following way.

#### Device Types

Each Device type will have a list of user names and RBAC permissions associated with it. If the user is not in the list, he is only allowed to manipulate Device types if he has the *WriteDeviceTypes* permission. If his user name or one of his RBAC permission is in the list, then the user can **edit** the Device type (change its name and description), and also **add**, **edit** and **delete** any of the properties, artifacts and tags associated with the Device type.

The user is also allowed to modify the authorization list of the Device type in any way. Removing himself (or all users) from the list will prevent him from manipulating this Device type in the future.

Each property associated with the Device type will also have an authorization list. The users authorized by that list are able to modify the value of the property and also remove themselves from the authorization list (in case the user is listed by its user name).

#### Devices

Each Device will have a list of user names and RBAC permissions associated with it. If the user is authorized by the list, he is able to change the Device serial number and its description. In addition to that the user is also allowed to **add**, **edit** and **delete** any of the properties, artifacts and tags associated with the Device. In accordance with the UI the operations on the properties are only limited to changing and deleting their value.

#### Containers and Installation Slots

Each Container and Installation slot will have a list of user names and RBAC permissions associated with it. If the user is authorized by the list, he is able to change the Container and Installation slot name and its description. In addition to that the user is also allowed to **add**, **edit** and **delete** any of the properties, artifacts and tags associated with the Container and Installation slot. In accordance with the UI the operations on the Installation slot properties are only limited to changing and deleting their value.

In addition the user being authorized by the authorization list on the current entity, he is also granted the same permissions if he is authorized by any list of the entities that are ancestors of this node in the *contains* hierarchy. In practice this means that granting a user permissions on some Container this also grants him permissions on all Containers and Installation slots that are children of this container.

## UI Layer

The application UI will be implemented using Java EE 7 JSF technology for building rich user interfaces in a web browser. There are 4 broad aspects of the UI (CCDB\_029, CCDB\_190, CCDB\_197):

* Defining data that will be put into the CCDB (configuration)
* Managing data in the CCDB
* Inspecting data in the CCDB
* Generating reports

A super user can define device types and properties that can be used by the users.

Authorized user can add device instances based on definitions, change the device instance properties and associate additional properties to the device instance.

### Data Definitions

Before users can start using the application, the super user needs to provide various basic data definitions that the users can use. This is achieved through following menus titled:

* Units
* Enumerations
* Properties
* Device types

#### Units

Units are used by various device properties. The main goal of providing a predefined set of units is to prevent proliferation of units in the system, which increases the chance for user errors, i.e. different users could use different units (W vs. mW) for the same property. The unit definitions can be reached through the Units menu.

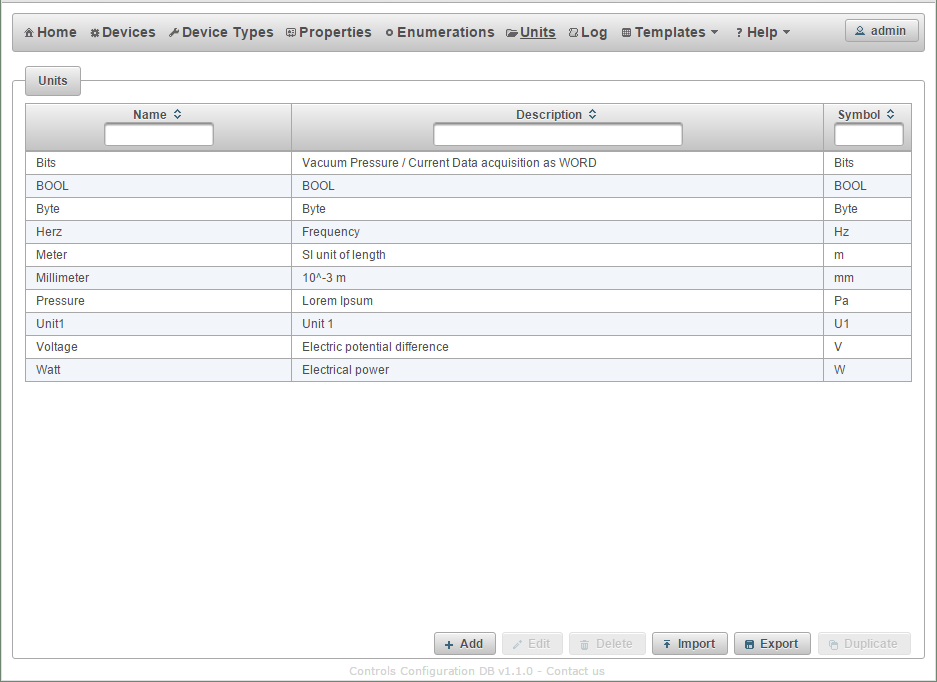


Figure 10 Unit definitions

The actions available to the super user are:

* Add a new unit
* Edit an existing unit
* Delete a unit
* Import units from an Excel file
* Export units to Excel or CSV file
* Duplicate selected units

Pressing the Add or the Edit button opens the unit definition dialog.

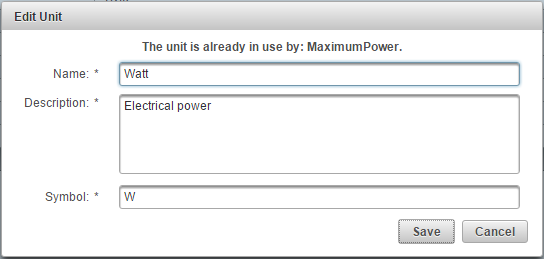


Figure 11 Unit definition

When saving a new unit, the CCDB application checks that no other unit with the same name exists. If it does an error message is displayed to the user.

The super user is only able to delete units that are not already used in the application.

See also section 3.1.6.1.

#### Enumerations

The super user must provide definition of all the enumerations that will be used in the CCDB. This can be done from the Enumerations screen.

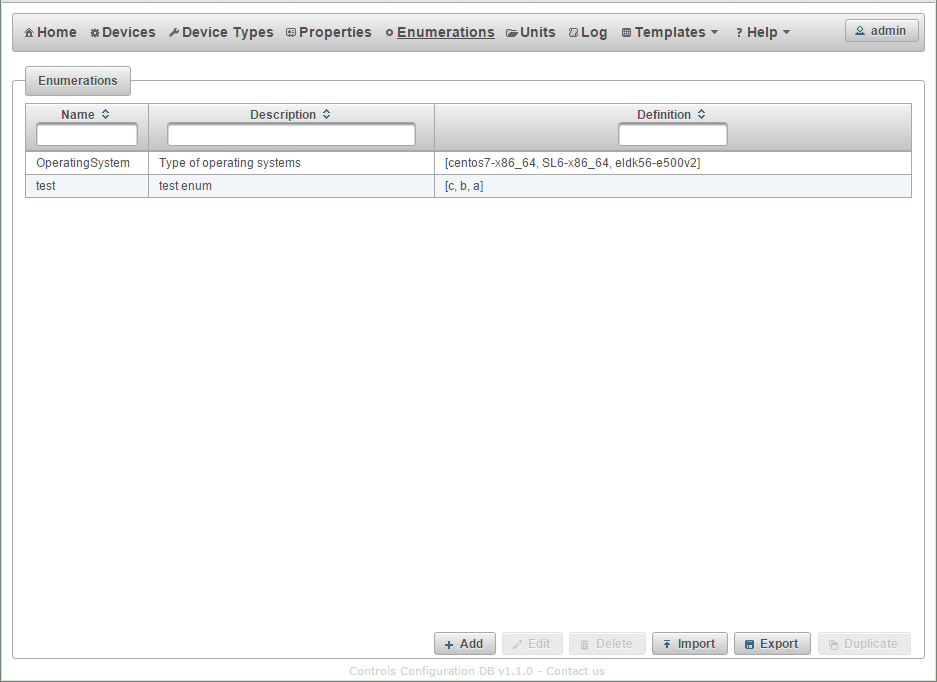


Figure 12 Browse enumerations screen

The actions available to the super user are:

* Add a new enumeration
* Edit an existing enumeration
* Delete an enumeration
* Import enumerations from an Excel file
* Export enumerations to an Excel or CSV file
* Duplicate selected enumerations

Pressing the Add or the Edit button opens the enumeration definition dialog.

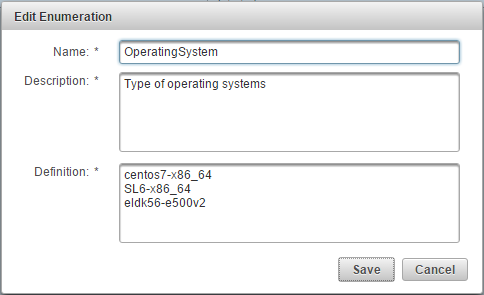


Figure 13 Defining enumeration

When saving enumeration, the CCDB application checks whether an enumeration with a similar name already exists (for new enumeration). If yes, an error message is returned. The enumeration values are entered one per line, each line contains a text representation displayed to the user.

When editing enumerations, the super user can change or remove the existing values of the enumeration only if the enumeration is not already used in the CCDB. Otherwise the super user can only add new values to the definition or change the order of the existing ones.

See also section 3.1.6.2.

#### Properties

After defining units, various properties can be defined. The properties are name/value pairs (CCDB\_090, CCDB\_106, CCDB\_170, CCDB\_110, CCDB\_111) that can be associated with the other application entities:

* Device types
* Containers
* Installation slots
* Device instances

Properties have a name, a description, a data type, an optional unit (CCDB\_170), and a uniqueness definition. The list of properties and their values is then displayed to the user when whenever he views a CCDB entity like container or device instance. Each CCDB entity can only have one property of each type associated with them. Properties are identified by name, meaning that CCDB cannot contain two property definitions with the same name. The property definitions can be accesses through the Properties menu.

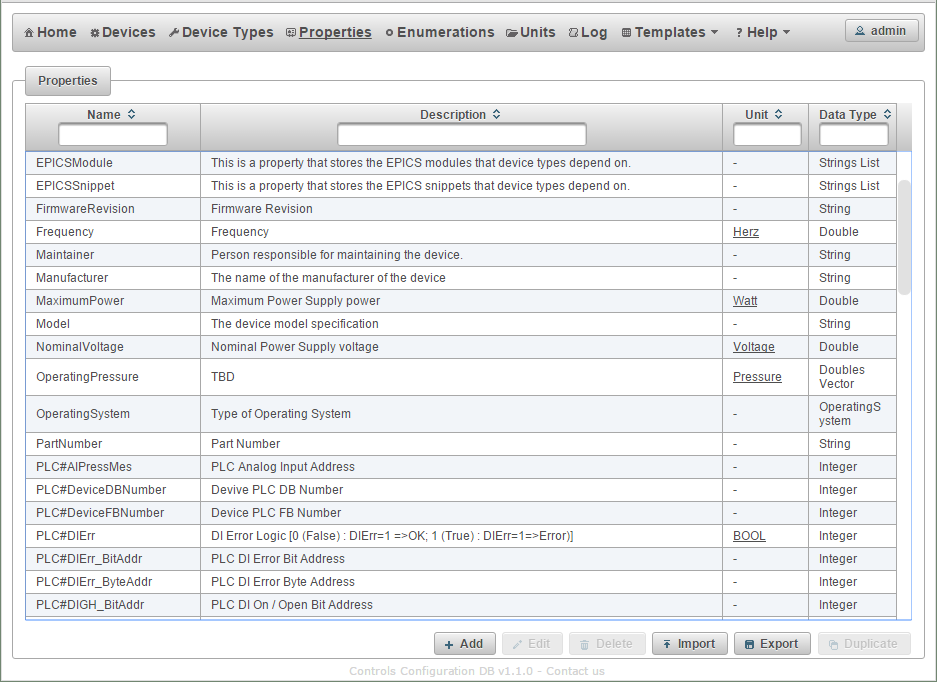


Figure 14 Browse properties screen

The actions available to the super user are:

* Add a new property
* Edit an existing property
* Delete a property
* Import properties from an Excel file
* Export properties to an Excel or CSV file
* Duplicate selected properties

Pressing Add or the Edit button brings the user to property definition screen.

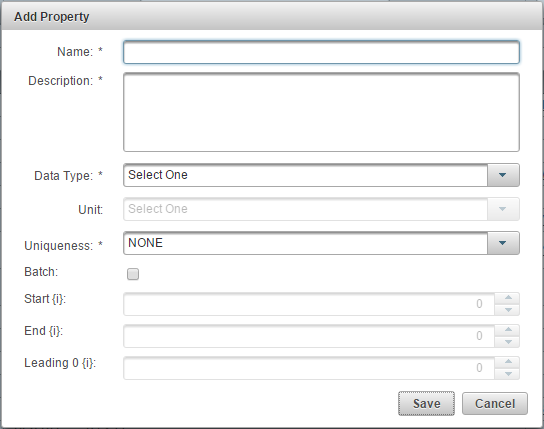


Figure 15 Property definition screen

On this screen the user can the property. Whether a unit can be selected depends on the Data Type. Property uniqueness defines how many properties of the same type (name) can be set to the same value. Possible values for uniqueness are:

* UNIVERSAL; no properties of the same type in the CCDB can be set to the same value
* TYPE; no properties of the same type assigned to the same device type can be set to the same value. The restriction does not apply for the same property defined for different device types
* NONE; the property value is not checked.

When defining a property, the user can choose to create multiple properties with the same name, but with a different number. The super user can define the starting and the ending index, as well as a number of leading zeros for the starting number. The super user must use the string {i} in the property name, to mark the place where the number will appear. The same marker can also be used in the property description.

#### Device Type

The users will fill CCDB with an inventory of all relevant devices that play a role in the ICS, and they can only add instances of devices and installation slots for which the device type was defined. The device types are defined by the super user through the Device Types menu.

The screen offers to Add or Edit (CCDB\_010, CCDB\_020, CCDB\_025, CCDB\_027, CCDB\_028) an existing device type. The screen also contains an action for manipulating the properties associated with each device type (CCDB\_010, CCDB\_020, CCDB\_025, CCDB\_027, CCDB\_028).

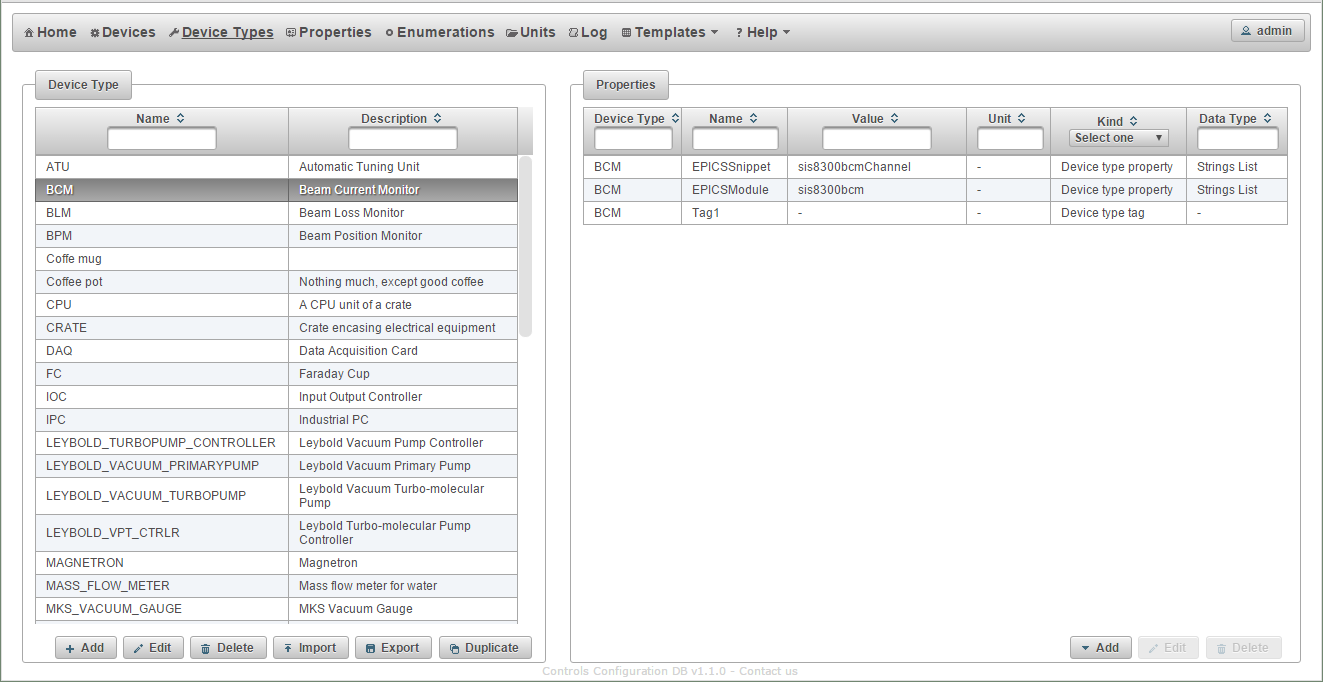


Figure 16 Device type listing

The super user can also filter the list of the device type by a string contained in the device type name or its description.

The removal (CCDB\_010, CCDB\_020, CCDB\_025, CCDB\_027, CCDB\_028) of the device type is only possible if there are no device instances or installation slots of this device defined in the CCDB application.

The actions available to the super user are:

* Add a new device type
* Edit an existing device type
* Delete a device type
* Import device types from an Excel file
* Export device types to an Excel or CSV file
* Duplicate selected device types

Pressing the Add button brings up the dialog for adding a new device type.

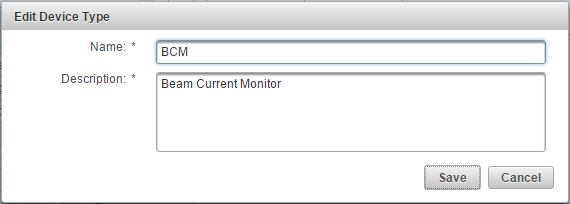


Figure 17 Define device type

#### Device Type Properties

After the device type has been defined, the super user can assign properties to it. This is achieved by clicking the Add button in the *Properties* part of the screen.

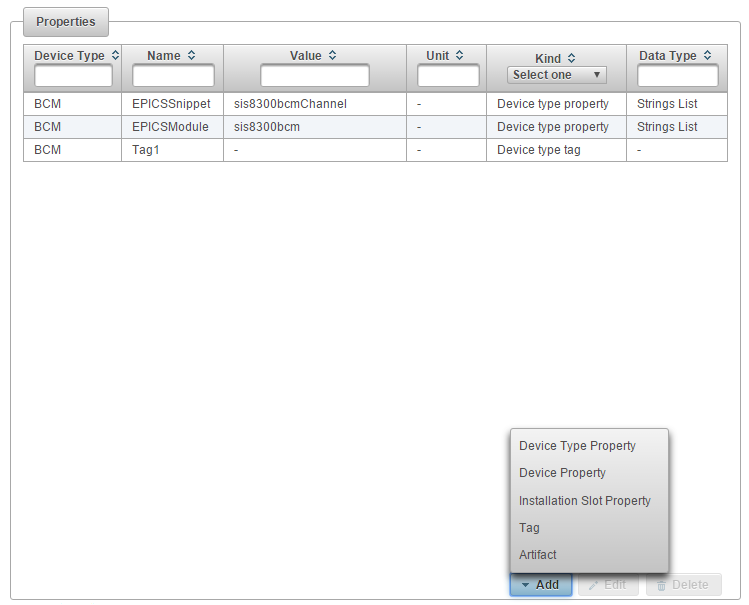


Figure 18 Defining device type properties

The list of available properties lists only the properties that can be associated with the device type or device instance.

CCDB device types will have three kinds of values associated with them:

1. *Device type properties*   
   properties that contain the same value for all device instances of this type. One example is device documentation and other similar items (CCDB\_160, CCDB\_106). These are any properties that can be associated with the device type.
2. *Device properties*  
   these are properties that hold different values for each device instance. Each device instance can only have property values that have been specified for device type by the super user.
3. *Installation slot properties*these are properties that hold different values for each installation slot. Each installation slot can only have property values that have been specified for the device type by the super user.
4. *tags*see section 3.1.7.4
5. *artifacts*see section 3.1.7.2

The device type properties are set/modified from the device type definition screen. The properties defined by *property definition* are set on the device instance screen. The values of properties of both types can be seen at the device instance screen.

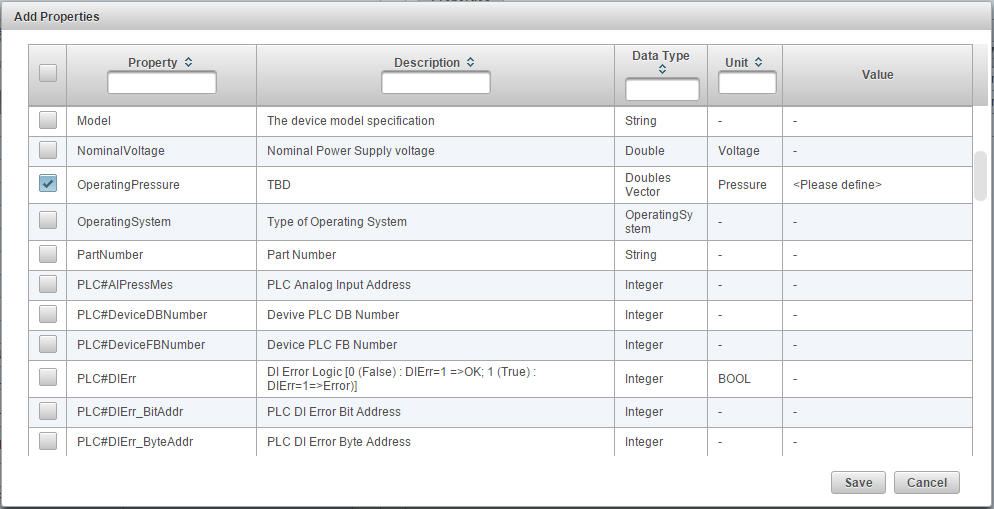


Figure 19 Adding device type property

When adding device type properties the super user is able to create multiple property values at the same time. The device type properties are set only once by the CCDB super user meaning that authorized users cannot redefined such a property at the device type level. Super user can still modify the property.

The super user is presented with an identical screen when defining device instance or installation slot properties. For those type of properties, the super user can define a default value that is used when a device or installation property is created. The super user can also choose not to select a default value.

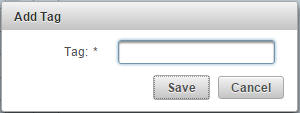


Figure 20 Adding or editing a tag

A tag is a simple text value associated with a device type. A device type tag can only be defined by a super user.

#### Containers

For some devices in the CCDB it will be important to mark their location inside some container. The containers can be either virtual or physical. The example of a virtual container is *subsystem* or the *ICS* group. An example of a physical container is a *rack* or *room*. Devices of different types are not physical containers. To accommodate this, the application will offer a way to define containers to which devices or other containers can then be assigned to (CCDB\_150).

The actions available to the authorized user are:

* Add a new container or installation slot
* Edit an existing container or installation slot
* Delete a container or installation slots
* Import the hierarchy from an Excel file
* Export the hierarchy to an Excel or CSV file
* Move the selected item up or down in the tree on the same level
* Copy or move selected containers or installation slots
* Expand or collapse the selected node

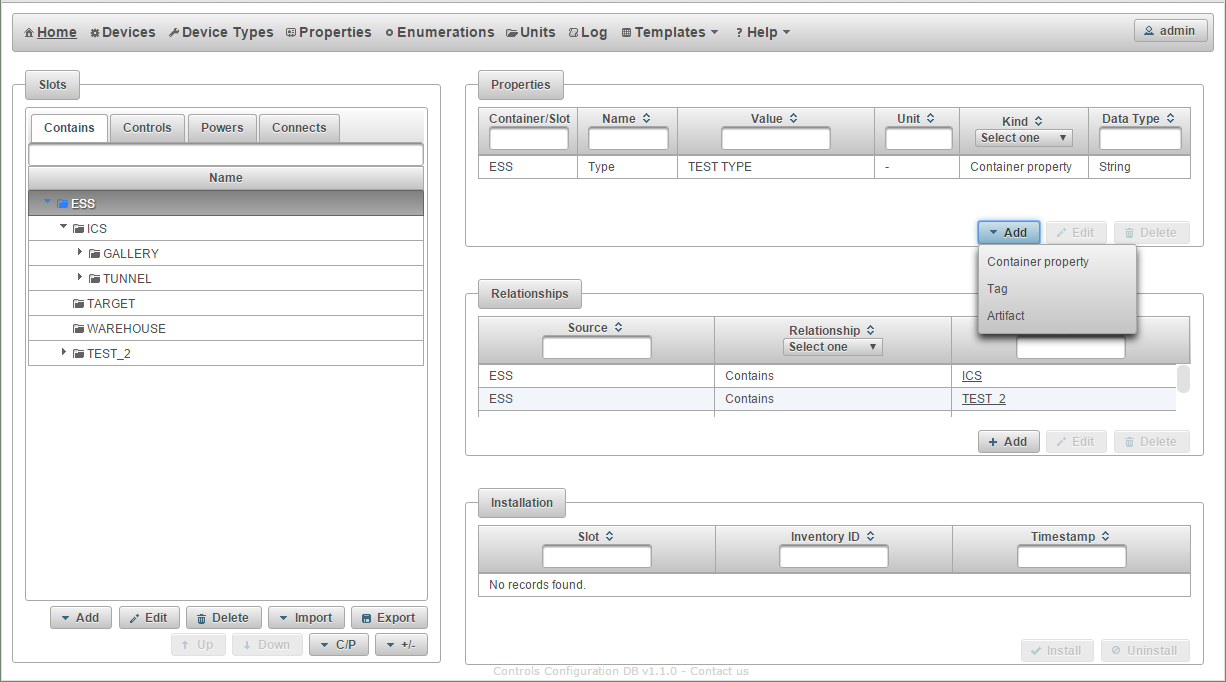


Figure 21 Browse containers

Container does not have many attributes besides name and an optional parent container (Figure 22) but additional properties can be assigned to it (Figure 21). A container without a parent is a root of some hierarchy.

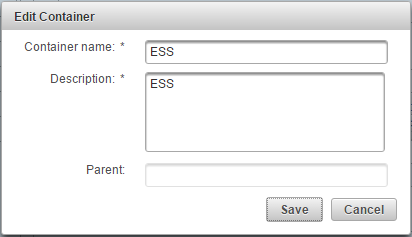


Figure 22 Define container

Each container can have any properties and tags associated with it. They can only be added and edited by the authorized users, but their values are displayed to the user when browsing appropriate hierarchy.

#### Installation Slots

Installation slots are places that will host device instances. The installation slots are the actual bearers of the information relevant to the ICS. They define the device type that needs to be installed in the slot, the naming convention name (if required) of the device to be installed and connection to other devices. The data about the name and connection is associated with the slot because it is relevant to the role and position of the device and not to the device instance itself. The device instance may get replaced, but the replacement will play the same role, when installed into this slot.

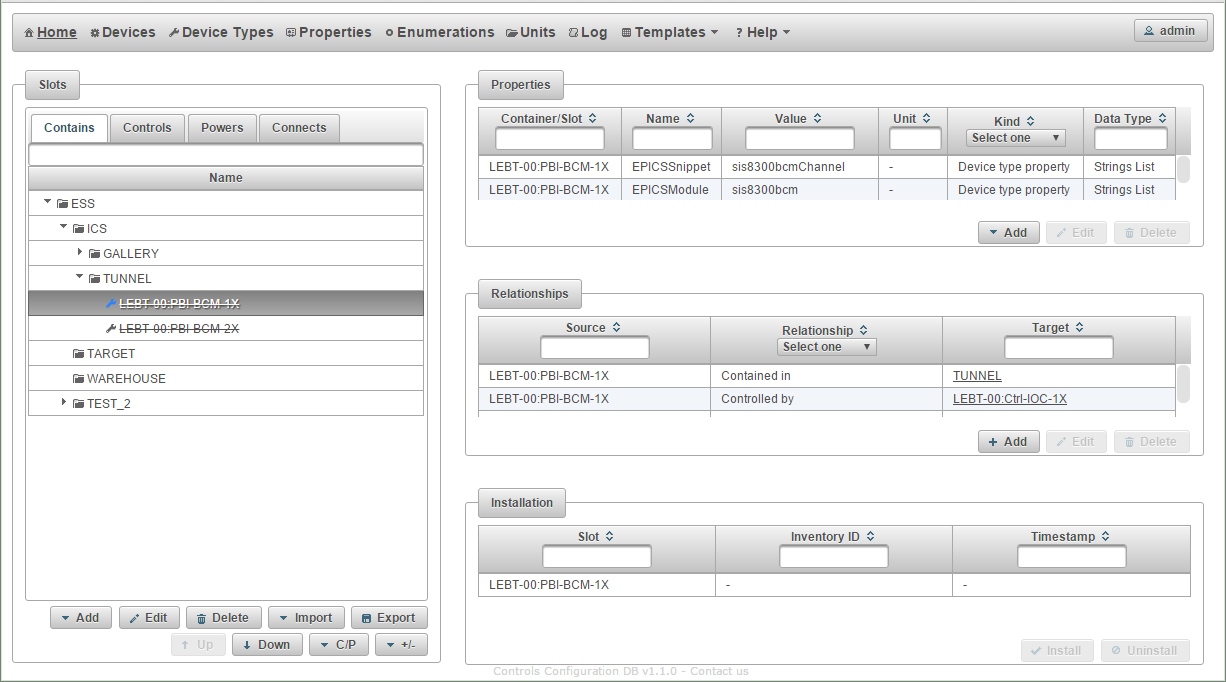


Figure 23 Browse installation slots

Pressing the Add button on a select container or slot adds a new installation slots to the selected container. On the new installation slot screen the user selects the type of the device that can be installed in the slot, the “naming convention” name of the slot and the description. Please note that the device name in the screenshot above is crossed out because the integration with the naming system was name could not be verified.

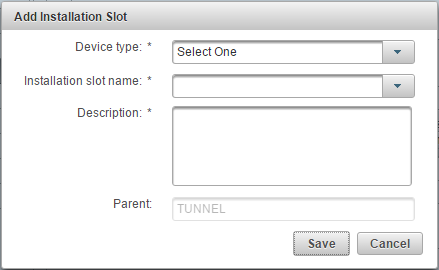


Figure 24 New installation slot

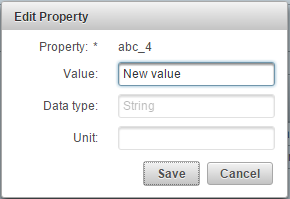


Figure 25 Installation slot properties

By adding container and installation slots in such a way, the user builds a hierarchy using the *contains* relationship.

The installation slots can be in relationship to other containers and installation slots in the database, which is manipulated in the *Home* screen (Figure 23, Figure 26). The *Relationships* table also displays all the cable connections between the installation slots the information for which is obtained from the Cable Database (CCDB\_103).

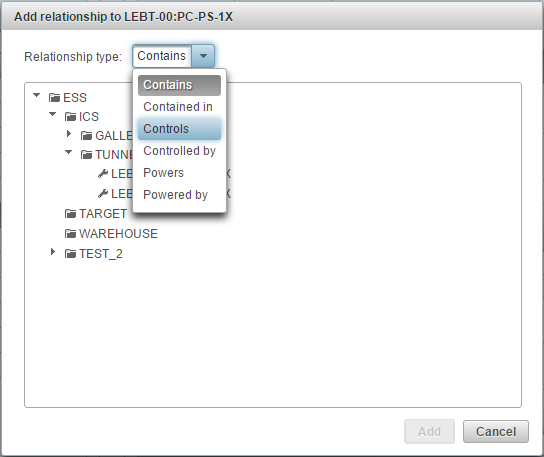


Figure 26 Adding relationships to the installation slots

The user can also select multiple container and installation slots in the tree hierarchy. The tables on the right display combined information for all the selected entities.

#### Device Installation

Into an empty selected installation slot the user can install devices selecting it again in the *Installation* table and pressing the *Install* button. Selection in the Installation table is necessary because that table can contain multiple installation slots if the user has selected so in the hierarchy tree.

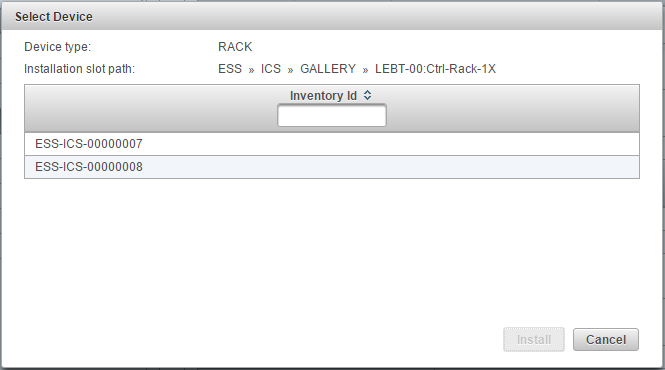


Figure 27 Device installation

The user is presented with the list of all devices of an appropriate device type that have not been installed yet. The user can then select a device instance and press the *Install* button.

### Adding and Browsing Device Instances

After the database has been populated by all the data definitions, the users can start defining various device instances. The device instances can be imported in bulk from the Excel spreadsheet, or the users can add devices from the user interface in the browser.

#### Device Types

Users can browse the device types defined in the CCDB by going to the Device types screen. On the left is displayed a list of all the defined devices, and filtering by the device type gives you a list of all device instances of that type.

The actions available to the authorized user are:

* Add a new device instance
* Edit an existing device
* Delete a device
* Import devices from an Excel file
* Export the devices to an Excel or CSV file
* Duplicate the selected devices

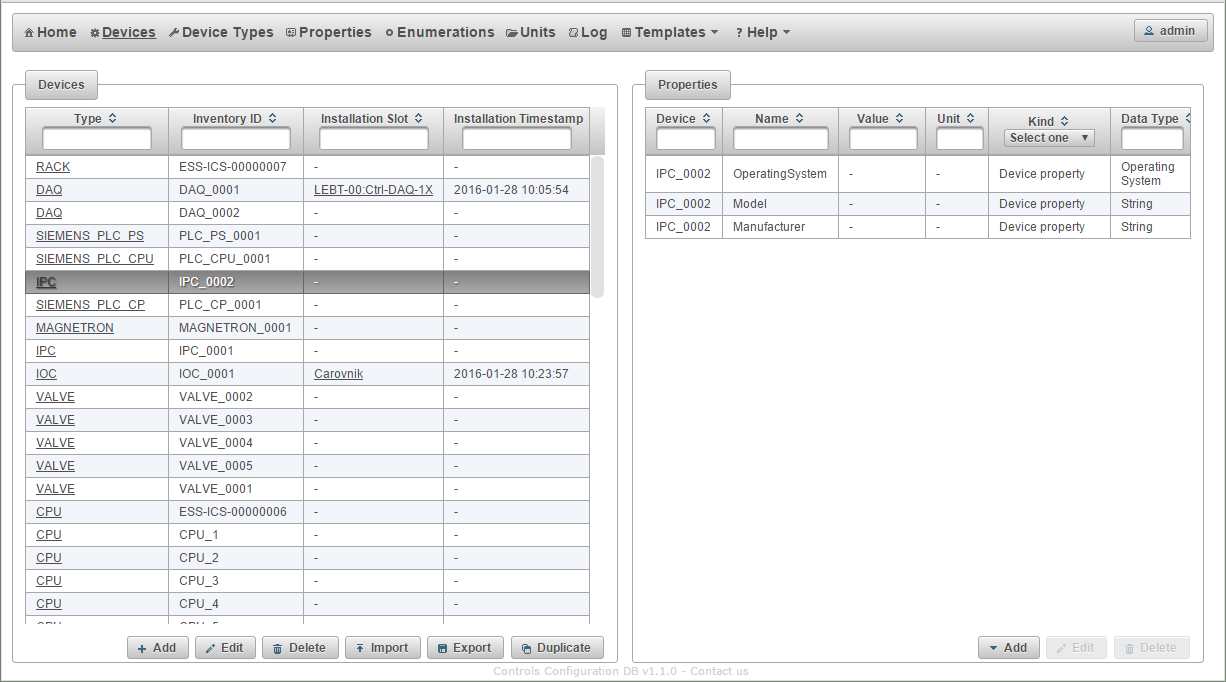


Figure 28 Listing device instances

The screen gives authorized users options to add, modify and delete (CCDB\_010, CCDB\_020, CCDB\_025, CCDB\_027, CCDB\_028) device instance records.

#### Adding Device Instance

Pressing the Add button opens a dialog for adding a new device instance to the database.

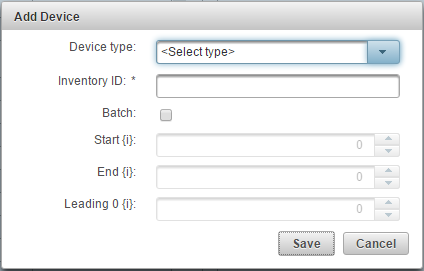


Figure 29 Adding a new physical device (CCDB\_060, CCDB\_070, CCDB\_073, CCDB\_110)

Each device must have a unique inventory ID (CCDB\_040, CCDB\_041, CCDB\_042), which is not limited to numerical characters only.

Similar to properties the user can choose to create multiple devices. The user can define the starting and the ending index, as well as a number of leading zeros for the starting number. The user must use the string {i} in the inventory ID, to mark the place where the number will appear.

A device instance can either be installed in the control system or not (spare, test equipment, in service).

#### Device Instance Details

The device details screen is shown in **Error! Reference source not found.**. The screen shows basic device information, which is similar for all devices.

The right table shows all the properties (CCDB\_085, CCDB\_107, CCDB\_110, CCDB\_111) and tags for this device. The following kinds of properties, artifacts and tags are possible for each device instance:

| Table 16 Device property types | |
| --- | --- |
| Property type | Description |
| Device type property | Identical for all devices of the same type. Such properties cannot be edited or removed. |
| Device property | Defined at the device type level, but different for each instance. Such properties cannot be removed. |
| Installation slot property | The table lists all the defined installation slot properties. In case the device is installed it shows their actual values as well. |
| Device type tag | A tag associated with a device type. Such tags cannot be edited or removed. |
| Device tag | A simple text value associated with a device instance. Such tags can be removed. |
| Installation slot tag | Installation slot tags are only shown for installed devices. |
| Device type artifact | An artifact associate with the device type. Such artifacts cannot be edited or removed. |
| Device artifact | An attachment or an URL reference associated with the device instance. Such artifacts can be edited or removed. |
| Installation slot artifact | Installation slot artifacts are only shown for installed devices. |

The user can only edit the device properties which have been defined at the device type level. The user can however add device instance tags and artifacts.

### Importing Data

To import data in bulk, and also to take advantage of the fact that data about many systems already exists in many user generated spreadsheets, the CCDB application offers a way to import the data from an Excel worksheet directly into the database.

CCDB application offers two types of data import based on the privileges of the user. A super user can import data definitions (enumerations, units, property definitions, device types) and the data about the device instances. An authorized user can only import data about device instances.

The import functionality can work with Excel (CCDB\_198) files for import. The Excel files may contain many worksheets, but the import function only looks for data in the first one. All additional worksheets found in the file are ignored.

#### Data Import

The type of data the user can import depends on the screen the user is on (see Figure 10, Figure 12, Figure 14, Figure 16, Figure 21, and Figure 28). In the beginning the user sees File input control, a Choose button (Figure 30). The user can select the Excel spreadsheet file from his disk by clicking on the Choose button. The user can also download sample worksheet files that he can fill with his own data from the *Templates* menu (see Figure 28).

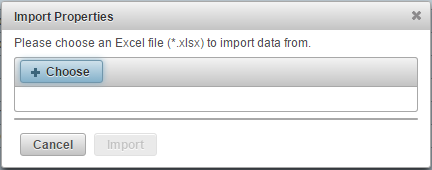


Figure 30 Data import dialog

The file is imported and the success of operation is shown to the user. In case of errors the entire file is rejected, and the errors are reported to the user.

In case the user is changing the relationships, the file can contain only relationship additions or deletions. The file is rejected if the relationship deletion would leave a container or installation slot orphaned. In such case an error describing the problem is displayed to the user.

#### Excel File

To import or update bulk data it is possible to import the Excel files (CCDB\_198) into the CCDB application. CCDB application super user can import the following types of data into the CCDB application:

* Enumerations
* Units
* Properties
* Device types
* Containers
* Installation slots
* Physical devices

For import each type of data needs to be in its own spreadsheet file. Upon import the user needs to select the type of data that is being imported. If the Excel file contains multiple worksheets, the application expects the data in the first one; other spreadsheets are ignored. The first column in each workbook specifies a command to the CCDB application instructing it what the appropriate line contains. The correct commands for each type of data are defined by the Excel template.

If the first column of the line does not contain a value, then this line is ignored. The user can put any text, values and formulas into this line. In effect this is a *comment* line of the import file. Also, the user can put any sort of data into the empty cells to the right of the data table.

The *create* family of commands instructs the CCDB application to create a new instance of the appropriate data record. The data record is identified by the value of the *name* column.

The *update* family of commands instructs the CCDB application to update an already existing data record. The data record is identified by the value of the *name* column.

The *delete* command instructs the CCDB application to delete the data import identified by the *name* column. If such record does not exist the error is reported. The information in all other columns is ignored and can be omitted.

If the Excel file contains any other type of data and not the device instance data, the import procedure will still try to interpret it as the device instance data. In case this is possible, it can lead to unexpected results.

### Generating Reports

The report generator (CCDB\_197, CCDB\_190) is started by pressing selecting the Reports action from the menu. If opens a screen shown in Figure 31.

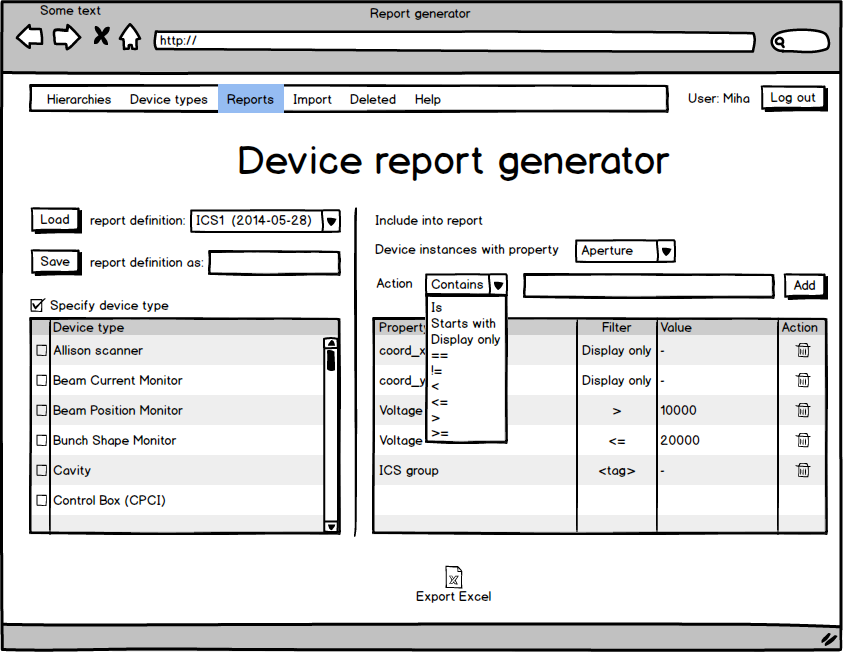


Figure 31 Report generation screen

The user can list the record fields and properties included in the report. The report will only include properties listed in the in the table on the right. For each property the user can select the value which decides whether the device instance is part of the report or not. It is also possible to select properties which are included in the report regardless of their value. The properties that are not selected are not part of the report. This gives users the options to only include information that they find relevant. In the table on the right side of the screen it is possible to limit the report based on the type of the device.

Pressing on the Excel icon at the bottom of this screen triggers the report generation based on the criteria specified.

After defining all the fields and filters for the report the user can save this definition so it can be reused at a later time. When saving a new definition the user must specify a name. If a report definition with such a name already exists, the user has to confirm replacement of the old report definition with the new one. The list of stored reports will also show the date the report definition was last modified.

### Data Export

While the database backup is in the domain of the database server the users will also have the option to export all the data in the CCDB into an Excel file (CCDB\_180). The main purpose of this export is to enable data migration and backup during the early stages of the project, when the server infrastructure may still not be complete and functioning properly. The export functionality is available to all users with appropriate RBAC permission (any user that is able to read the data).

### Deleted Entries

After the various entities get deleted from the CCDB application, the audit log entities remain in the database, but can no longer be associated with the deleted entity. The audit log entries can be found in the *Log* menu. Here the users can access information on all the actions in the CCDB application.

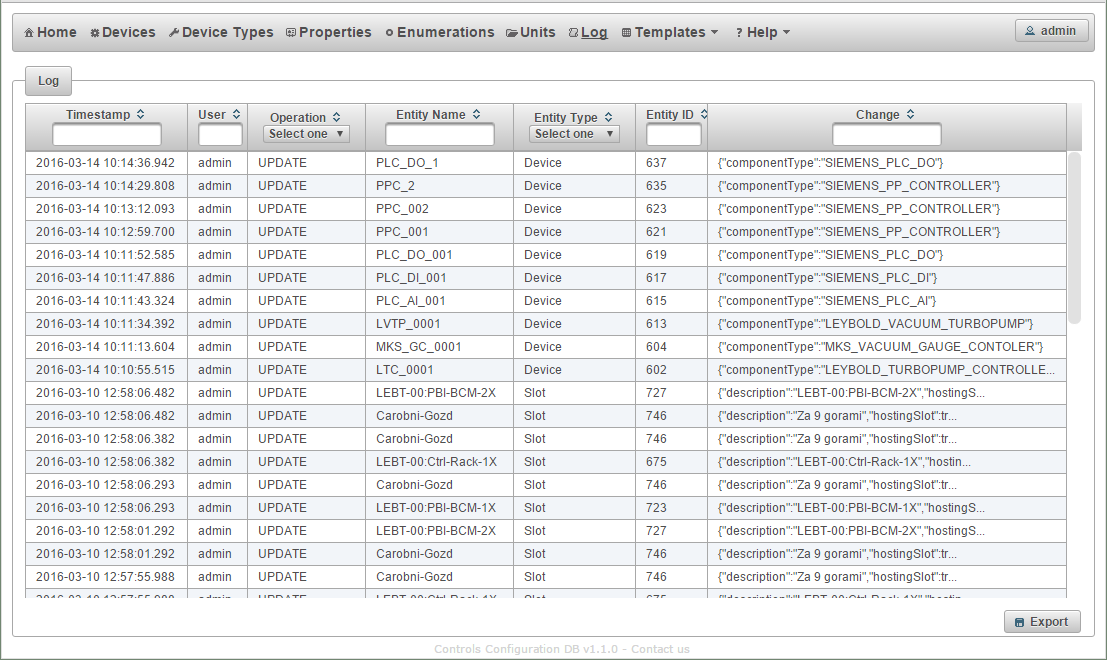


Figure 32 Audit log entries

On the left side the user sees a list of all entries initially sorted by the action time. The user can select some other column for sorting or type a name of the entity to filter the list.

Selecting a deleted entity displays a log of actions for this entity. To get a detailed information on a specific log entry, the user can click on the *change* text for the desired log entry. Log entry details are shown in Figure 33.



Figure 33 Audit log entry details

## Service Layer

The service layer will provide business logic for the application.

At the minimum it will need to perform the following tasks:

* Integrate with RBAC to enforce correct behavior based on user authorization
* Take care of the audit history for all entity operations
* Integrate with naming and cabling in appropriate ways

The service layer will take care of things like viewing, creating and modifying entities. It will need to work especially carefully with hierarchies. Hierarchical organization of data has extremely high impact on the data consistency. This is even more complicated if the same data can be shown in different disjoint hierarchies. Depending on the nature of data it may be rather complicated to ensure data consistency even within one hierarchy, but if the same data is presented in multiple hierarchies, this issue becomes much harder.

### RESTful Interface

To start with the CCDB application will offer a read-only RESTful interface (CCDB\_194). The RESTful interface will support access to device types, containers, installation slots and device instances.

#### Device Types

The RESTful service under the URL <http://server/rest/deviceTypes> will return a list of all device types in the CCDB application.

The URL [*http://server/ccdb/rest/deviceTypes/<name>*](http://server/ccdb/rest/deviceTypes/%3cname%3e) will return all the information about a specific device type.

Artifact attachments from specific device type can be downloaded through URL [http://server/rest/deviceTypes/<deviceTypeName>/download/<fileName>](http://server/rest/deviceTypes/%3cdeviceTypeName%3e/download/%3cfileName%3e).

##### Examples

URL: <http://server/rest/deviceTypes>

<deviceTypes xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <deviceType>  
 <name>CPU</name>  
 <description>CPU</description>  
 <artifacts>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 </artifacts>  
 </deviceType>  
 <deviceType>  
 <name>CPU\_1</name>  
 <description>CPU</description>  
 </deviceType>  
 <deviceType>  
 <name>CRATE</name>  
 <description>Crate</description>  
 </deviceType>  
 ...  
</deviceTypes>

Figure 34 Sample RESTful service output listing all device type

URL: <http://server/rest/deviceTypes/CPU>

<deviceType xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <name>CPU</name>  
 <description>CPU</description>  
 <artifacts>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 </artifacts>  
</deviceType>

Figure 35 Sample RESTful service output showing a specific device type

To download the README.txt artifact file the following URL can be used: <http://server/rest/deviceTypes/CPU/download/README.txt>

#### Installation Slots

The RESTful interface under the URL <http://server/rest/slots> will return a list of all the installation slots in the CCDB application. The URL supports the parameter *deviceType* and returns a list of all installation slots of specified device type.

For each installation slot the list will also contain a parent installation slot, a list of its children, the device type, properties and artifacts.

The URL [*http://server/ccdb/rest/slots/<name>*](http://server/ccdb/rest/slots/%3cname%3e) will return all the information about a specific installation slot.

The URL [*http://server/ccdb/rest/slot/<slotName>/download/<fileName>*](http://server/ccdb/rest/slot/%3cslotName%3e/download/%3cfileName%3e) will return artifact attachment file from the specified slot. If file with such name is not found, the service searches for a file with such name in the installed device (if any) and device type.

##### Examples

URL: <http://server/rest/slots>

<slots xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <slot>  
 <name>H1:3:1:DC07</name>  
 <description>DIO</description>  
 <deviceType>DigitalCard01</deviceType>  
 <artifacts>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 </artifacts>  
 <parents>  
 <parent>H1:3:1:VME01</parent>  
 </parents>  
 <children>  
 <child>H1:3:1:DIG02</child>  
 </children>  
 <controls>  
 <control>H1:3:1:DIG02</control>  
 </controls>  
 <poweredBy>  
 <powerBy>H1:2:1:DC06</powerBy>  
 </poweredBy>  
 <properties>  
 <property>  
 <name>TypeProp</name>  
 <value>-1.1</value>  
 <dataType>Double</dataType>  
 <kind>TYPE</kind>  
 </property>  
 <property>  
 <name>SignalName1</name>  
 <value>null</value>  
 <dataType>String</dataType>  
 <kind>SLOT</kind>  
 </property>  
 <property>  
 <name>SignalName2</name>  
 <value>null</value>  
 <dataType>String</dataType>  
 <kind>SLOT</kind>  
 </property>  
 </properties>  
 </slot>  
 ...  
</slots>

Figure 36 Sample RESTful service output listing all installation slots

URL: <http://server/rest/slots?deviceType=CPU>

<slots xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <slot>  
 <name>CPU1</name>  
 <description>Hello</description>  
 <deviceType>CPU</deviceType>  
 <artifacts>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 </artifacts>  
 <parents>  
 <parent>LEBT-00:Ctrl-Crate-1</parent>  
 </parents>  
 <properties>  
 <property>  
 <name>Manufacturer</name>  
 <value>Intel</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 <property>  
 <name>Model</name>  
 <value>i7</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 </properties>  
 </slot>  
 ...  
</slots>

Figure 37 Sample RESTful service output listing all installation slots of specified device type

URL: <http://server/rest/slots/CPU1>

<slot xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <name>CPU1</name>  
 <description>Hello</description>  
 <deviceType>CPU</deviceType>  
 <artifacts>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>SLOT</kind>  
 </artifact>  
 <artifact>  
 <name>NOTES.txt</name>  
 <description>Installation notes</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 </artifacts>  
 <parents>  
 <parent>LEBT-00:Ctrl-Crate-1</parent>  
 </parents>  
 <properties>  
 <property>  
 <name>Manufacturer</name>  
 <value>Intel</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 <property>  
 <name>Model</name>  
 <value>i7</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 </properties>  
</slot>

Figure 38 Sample RESTful service output showing a specific installation slot

The URL <http://server/rest/slots/CPU1/download/README.txt> will return the artifact attachment file *README.txt* from the installation slot *CPU1*.

The URL <http://server/rest/slots/CPU1/download/NOTES.txt> will return the artifact attachment file *NOTES.txt* from the device type *CPU*.

The artifact attachment file *README.txt* of the *CPU device type* cannot be downloaded through the installation slot but has to be downloaded from the device type directly.

#### Installation Slot Names

Since the installation slot related URLs return a large amount of information, there is also a special service that returns just the names of all the installation slots and containers in the database. The container names may be duplicated, since there is no unique requirement regarding their name. This is for intended for applications that may want to integrate with CCDB and would want to display a list of possible slots in the CCDB based on type.

##### Examples

URL: <http://server/rest/slotNames>

<names xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <name type="slot">H1:3:1:DC07</name>  
 <name type="slot">H1:1:1:DC02</name>  
 <name type="slot">H1:1:2</name>  
 <name type="slot">H1:3:2:DC08</name>  
 <name type="SLOT">H1:3:3:DC09</name>  
 <name type="SLOT">H1:1:2:DC03</name>  
 <name type="SLOT">H1:1:1:DC02:B01</name>  
 <name type="SLOT">H1:1:1:DC02:B02</name>  
 <name type="CONTAINER">DEMO</name>  
 <name type="SLOT">H1:2:1:DC05</name>  
 <name type="SLOT">H1:2:1:DC06</name>  
 <name type="SLOT">H1:2:2:DC07</name>  
 <name type="SLOT">H1:2:2:DC08</name>  
 <name type="CONTAINER">WAREHOUSE</name>  
 <name type="CONTAINER">GALLERY</name>  
 ...  
</names>

Figure 39 Sample RESTful service output listing all installation slot and container names

URL: <http://server/rest/slotNames?deviceType=CPU>

<names xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <name type="SLOT">H1:3:1:CPU07</name>  
 <name type="SLOT">H1:1:1:CPU02</name>  
 <name type="SLOT">H1:3:2:CPU08</name>  
 <name type="SLOT">H1:3:3:CPU09</name>  
 <name type="SLOT">H1:1:2:CPU03</name>  
 <name type="SLOT">H1:2:1:CPU05</name>  
 <name type="SLOT">H1:2:1:CPU06</name>  
 <name type="SLOT">H1:2:2:CPU07</name>  
 <name type="SLOT">H1:2:2:CPU08</name>  
 <name type="SLOT">H1:1:1:CPU02\_1</name>  
 <name type="SLOT">Cnt1:DIO:CPU0001\_1</name>  
 <name type="SLOT">H1:1:1:CPU02\_2</name>  
 <name type="SLOT">Cnt1:DIO:CPU0001</name>  
</names>

Figure 40 Sample RESTful service output listing installation slots of a specified device type

#### Device Instances

The RESTful interface under the URL <http://server/rest/devices> will return a list of all device instances in the CCDB application.

The URL [*http://server/rest/devices/<inventoryID>*](http://server/rest/devices/%3cinventoryID%3e) will return all the information about a specific devices. The information will also contain a list of all devices and containers that the device instance is in a relationship with.

The URL [*http://server/ccdb/rest/devices/<inventoryID>/download/<fileName>*](http://server/ccdb/rest/devices/%3cinventoryID%3e/download/%3cfileName%3e) will return artifact attachment file from specific device. If file with such name is not found, file from currently slot device is currently installed in or device type gets downloaded.

##### Examples

URL: <http://server/rest/devices>

<devices xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <device>  
 <inventoryId>ESS-ICS-00000001</inventoryId>  
 <deviceType>CRATE</deviceType>  
 <properties>  
 <property>  
 <name>Manufacturer</name>  
 <value>MegaCrates Corp.</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 <property>  
 <name>Model</name>  
 <value>TheCube-1337</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 </properties>  
 <artifacts>  
 <artifact>  
 <name>NOTES.txt</name>  
 <description>Installation notes</description>  
 <type>FILE</type>  
 <kind>DEVICE</kind>  
 </artifact>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 </artifacts>  
 </device>  
 <device>  
 <inventoryId>ESS-ICS-00000002</inventoryId>  
 <deviceType>CRATE</deviceType>  
 <properties>  
 <property>  
 <name>Manufacturer</name>  
 <value>MegaCrates Corp.</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 <property>  
 <name>Model</name>  
 <value>TheCube-1337</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 </properties>  
 </device>  
…  
</devices>

Figure 41 Sample RESTful service output listing all devices

URL: <http://server/rest/devices/ESS-ICS-00000001>

<device xmlns="https://europeanspallationsource.se/rest/ccdb">  
 <inventoryId>ESS-ICS-00000001</inventoryId>  
 <deviceType>CRATE</deviceType>  
 <properties>  
 <property>  
 <name>Manufacturer</name>  
 <value>MegaCrates Corp.</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 <property>  
 <name>Model</name>  
 <value>TheCube-1337</value>  
 <dataType>String</dataType>  
 <kind>TYPE</kind>  
 </property>  
 </properties>  
 <artifacts>  
 <artifact>  
 <name>NOTES.txt</name>  
 <description>Installation notes</description>  
 <type>FILE</type>  
 <kind>DEVICE</kind>  
 </artifact>  
 <artifact>  
 <name>README.txt</name>  
 <description>Readme file</description>  
 <type>FILE</type>  
 <kind>TYPE</kind>  
 </artifact>  
 </artifacts>  
</device>

Figure 42 Sample RESTful service output showing specific device

The URL <http://server/rest/devices/ESS-ICS-00000001/download/README.txt> can be used to download the artifact attachment file *README.txt* from the device *ESS-ICS-00000001*.

The URL <http://server/rest/devices/ESS-ICS-00000001/download/NOTES.txt> can be used download the artifact attachment file *NOTES.txt* from the device type *CRATE*.

The artifact attachment file *README.txt* of the *CPU* device type cannot be downloaded through the installation slot but has to be downloaded from the device type directly.

# Requirements Traceability

| Table 17 Requirements traceability | |  |
| --- | --- | --- |
| Number | Title | Realization (page) |
| CCDB\_010 | Control Boxes | 14, 40, 40, 40, 49 |
| CCDB\_020 | Machine Protection System (MPS) Devices | 14, 40, 40, 40, 49 |
| CCDB\_025 | Personal Protection System (PPS) Devices | 14, 40, 40, 40, 49 |
| CCDB\_027 | Infrastructure | 14, 40, 40, 40, 49 |
| CCDB\_028 | Other Devices | 14, 40, 40, 40, 49 |
| CCDB\_029 | Web-based User Interface for Add, Modify, and Delete | 33 |
| CCDB\_030 | Unique ID | 13 |
| CCDB\_040 | Naming Convention | 18, 50 |
| CCDB\_041 | Naming Convention Integration | 50 |
| CCDB\_042 | Naming Convention Integration | 50 |
| CCDB\_050 | Name | 17, 18 |
| CCDB\_060 | Serial Number | 49 |
| CCDB\_070 | Global Coordinates | 49 |
| CCDB\_073 | Orientation Angle | 49 |
| CCDB\_080 | Physical Size | 16, 21 |
| CCDB\_085 | Shape Description | 50 |
| CCDB\_090 | Rack Location | 16, 21 |
| CCDB\_103 | Cable Connection | 46 |
| CCDB\_106 | Water Connection | 16, 21, 37, 42 |
| CCDB\_107 | Firmware and FPGA | 16, 20, 21, 50 |
| CCDB\_110 | Tracking Status | 16, 15, 21, 37, 49, 50 |
| CCDB\_111 | Status type | 15, 21, 37, 50 |
| CCDB\_140 | History | 22, 22, 22, 23 |
| CCDB\_150 | Relationship To Other Devices | 14, 18, 43 |
| CCDB\_160 | Documentation | 42 |
| CCDB\_170 | Name Value Pairs | 16, 37 |
| CCDB\_175 | Access Control | 11, 30 |
| CCDB\_180 | Excel Export | 54 |
| CCDB\_190 | Reports | 33, 52 |
| CCDB\_194 | RESTful Interface | 55 |
| CCDB\_195 | API - Java | 11 |
| CCDB\_197 | Web-based User Interface for Reports | 33, 52 |
| CCDB\_198 | Import Format | 51, 51 |
| CCDB\_199 | Input Validation |  |
| CCDB\_200 | Size |  |

# References

1. Gysin S., Control System Configuration Database Requirements Document (1.9), ESS AB, 2013
2. J. Bobnar, Role Based Access Control Design Document (1.5), Cosylab, 2014
3. DISCS - Distributed Information Services for Control Systems, http://openepics.sourceforge.net/