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

**PROVAL MATRIX**

| Role | Name | Affiliation | Date |
| --- | --- | --- | --- |
| Author | Miha Vitorovič | Cosylab | 2014-07-30 |
| Reviewer | Suzanne Gysin | ESS AB |  |
|  | Timo Korhonen | ESS AB |  |
|  | Daniel Piso Fernández | ESS AB |  |
|  | Miha Reščič | ESS AB |  |
| Approver | Suzanne Gysin | ESS AB |  |
|  | Garry Trahern | ESS AB |  |

**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 |

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 Abbreviations 7

1. Introduction 8

2. Architecture 8

3. Design 10

3.1 Database 10

3.1.1 Change history 10

3.1.2 Data definitions 12

3.1.3 Device instances 12

3.1.4 Relationship 12

3.1.5 Device types 12

3.1.6 Data definition details 13

3.1.6.1 Units 13

3.1.6.2 Enumerations 14

3.1.6.3 Properties, tags and property values 14

3.1.6.4 Containers and installation slots 16

3.1.6.5 Device types 18

3.1.7 User data 19

3.1.7.1 Device instance 19

3.1.7.2 Artifacts 19

3.1.7.3 Property values 20

3.1.7.4 Tags 20

3.1.8 History control 21

3.1.8.1 Audited entities 22

3.1.8.2 Audit trail description 22

3.1.8.3 Examples 22

3.2 User roles and permissions 32

3.2.1 Configuration module authorization model 32

3.3 UI Layer 33

3.3.1 Data definitions 33

3.3.1.1 Enumerations 34

3.3.1.2 Units 35

3.3.1.3 Properties 37

3.3.1.4 Device type 39

3.3.1.5 Device type properties 42

3.3.1.6 Containers 45

3.3.1.7 Installation slots 48

3.3.2 Adding and browsing device instances 51

3.3.2.1 Device types 51

3.3.2.2 Adding device instance 52

3.3.2.3 Device instance details 53

3.3.2.4 Device installation 56

3.3.2.5 CCDB hierarchies 57

3.3.3 Importing data 59

3.3.3.1 Super user import 59

3.3.3.2 Authorized user import 61

3.3.3.3 Excel file 61

3.3.4 Generating reports 62

3.3.5 Data export 63

3.3.6 Deleted entries 63

3.4 Service layer 65

3.4.1 RESTful interface 66

3.4.1.1 Device types 66

3.4.1.2 Installation slots 67

3.4.1.3 Installation slot names 69

3.4.1.4 Device instances 71

4. Requirements traceability 71

5. References 72

List of tables

[Table 1 Possible “includes” relationship endpoints 17](#_Toc432067729)

[Table 2 History record description 21](#_Toc432067730)

[Table 3 Enumeration creation entry 22](#_Toc432067731)

[Table 4 Enumeration modification entry 23](#_Toc432067732)

[Table 5 Enumeration deletion entry 23](#_Toc432067733)

[Table 6 Unit creation entry 24](#_Toc432067734)

[Table 7 Property creation entry 24](#_Toc432067735)

[Table 8 Device type creation entry 25](#_Toc432067736)

[Table 9 Container creation entry 25](#_Toc432067737)

[Table 10 Adding relationships 26](#_Toc432067738)

[Table 11 Installation slot creation entry 27](#_Toc432067739)

[Table 12 Device type add property definition 28](#_Toc432067740)

[Table 13 Add device type property 29](#_Toc432067741)

[Table 14 Device type remove property definition 30](#_Toc432067742)

[Table 15 Relationship removal 30](#_Toc432067743)

[Table 16 Device type adding and removing tags 31](#_Toc432067744)

[Table 17 Device property types 54](#_Toc432067745)

[Table 18 Requirements traceability 71](#_Toc432067746)

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.

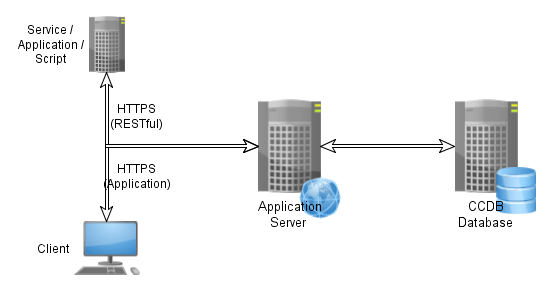


Figure 1 Three tier architecture

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.

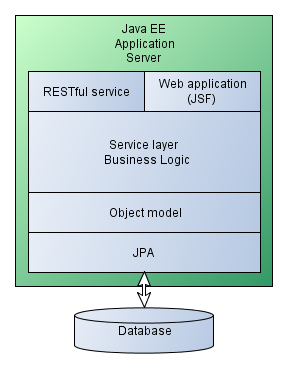


Figure 2 Java EE architecture

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

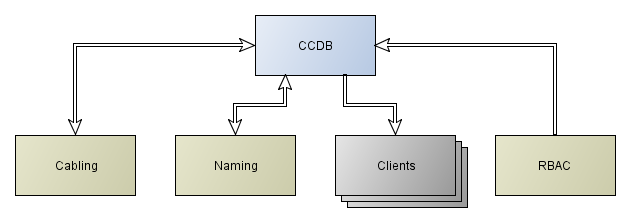


Figure 3 CCDB integration with other ICS services

# 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:

* Includes
* Controls
* Powers

The includes 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.

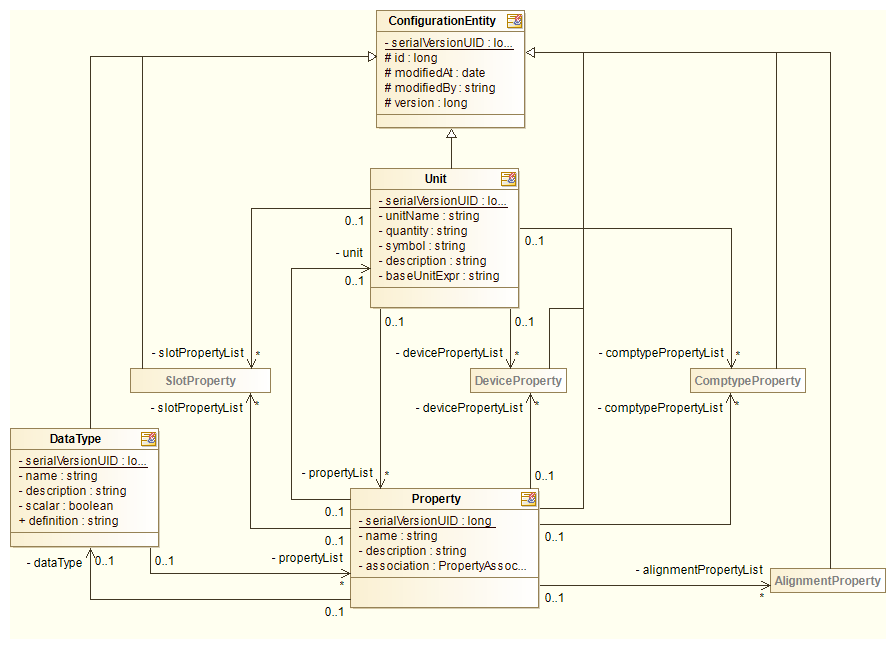


Figure 5 OM detail: Data type, Unit and Property

#### 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
* Quantity: Electric current
* 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)
* Association (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 4 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 association field specifies with which database entity the property can be associated. The property can be associated with a *device type* (available to super user only), *container*, *installation slot* or a *device instance*.

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

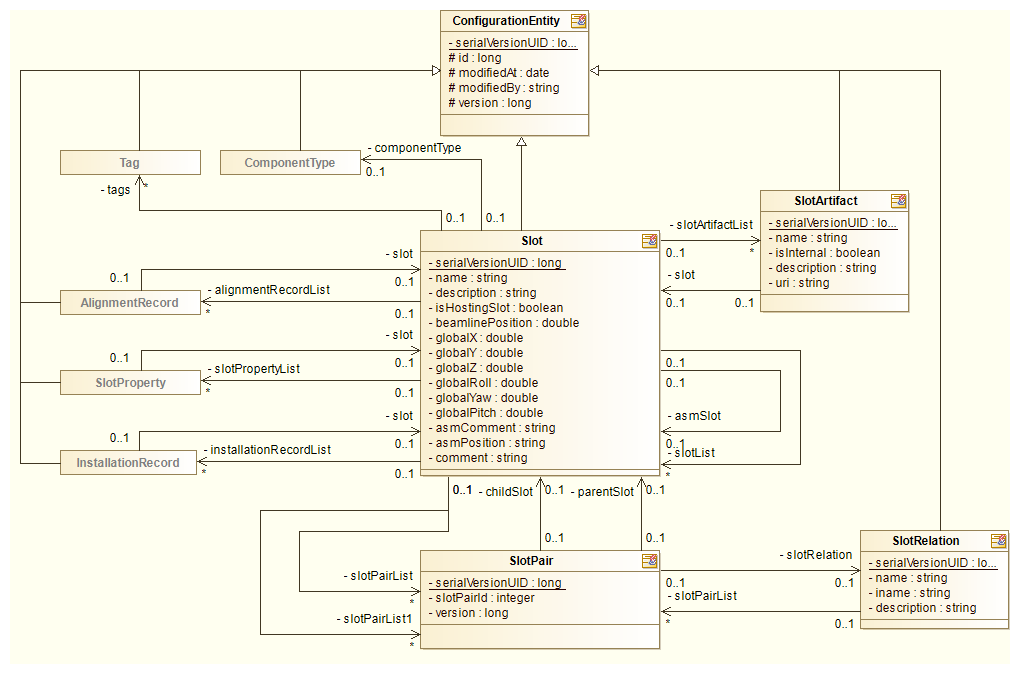


Figure 6 OM detail: Slot entity (containers and installation slots) (CCDB\_050)

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 *includes* relationship (CCDB\_150). The *includes* 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 *includes* relationship as a child. All other containers have a component type of \_GRP.

| Table 1 Possible “includes” 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 *includes* relationship as a parent. If yes, deletion has to fail. 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 implicitly, when installing device instances. 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

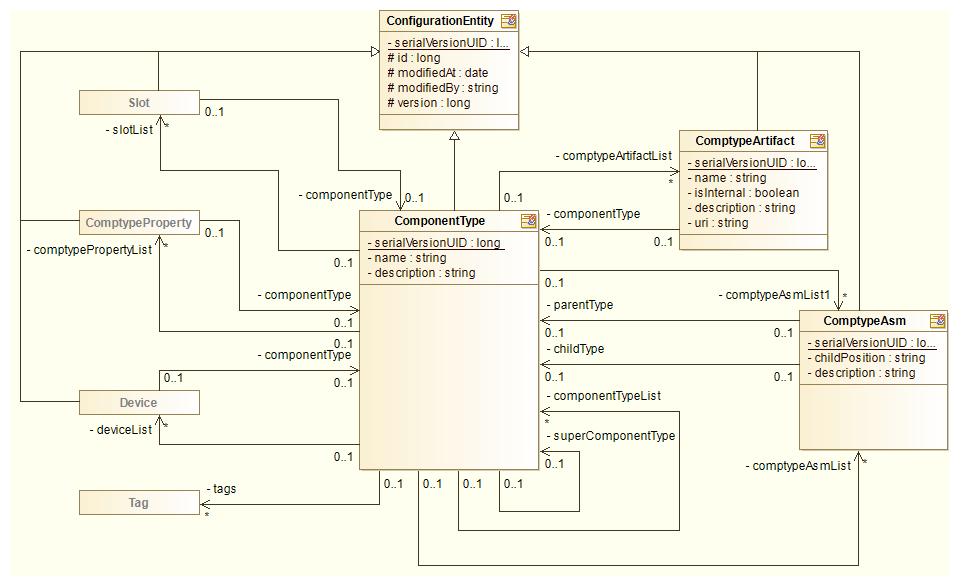


Figure 7 OM detail: device types

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 number of predefined fields 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, 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 have a value set for each such property. A property definition is a device type property without a value.

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.

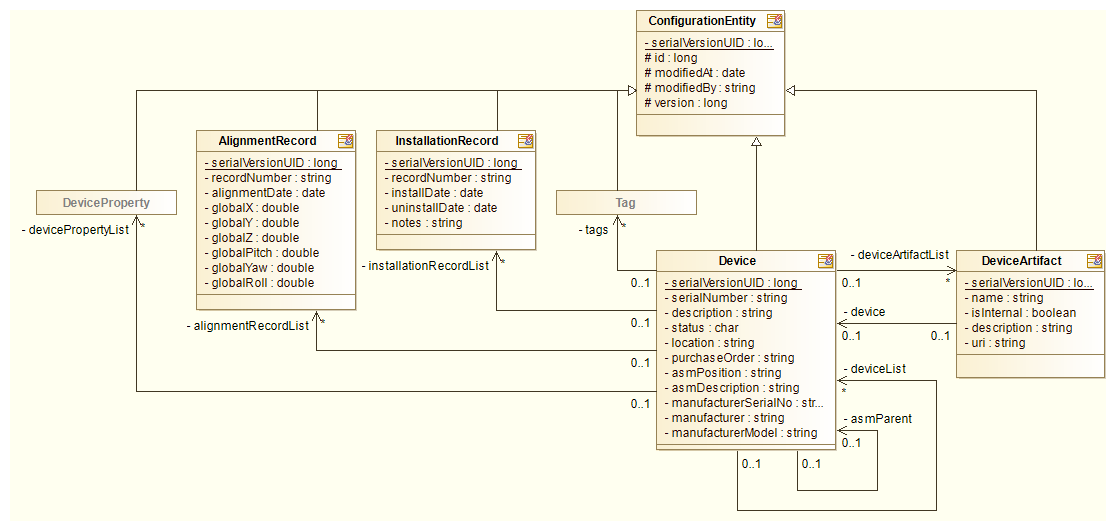


Figure 8 OM detail: device instances

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 can be 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 artifacts can be created for *device types* (super user only) or *containers*, *device instances* and *installation records* (authorized user).

**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

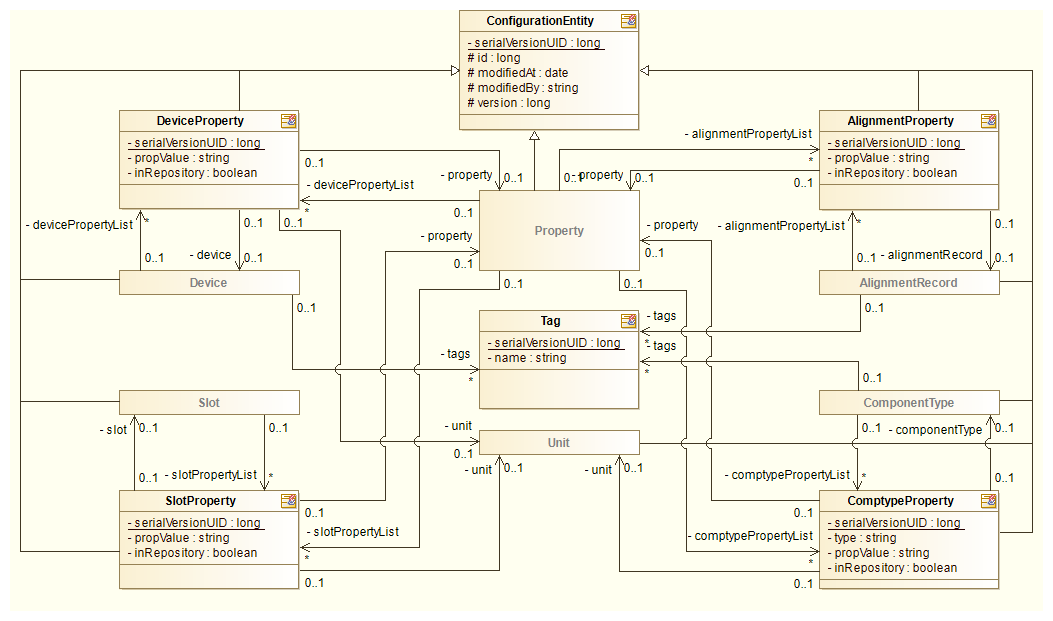


Figure 9 OM detail: Property values and tags

Property values are name value pairs associated with the database entity. The user is free to append new property values to a device instance as long as the *property association* (see 3.1.6.3) allows this. (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 if it contains less than 100 elements, otherwise the audit entry log will simply state *<changed>*.

1-D string array change will only be only be logged if the array contains less than 20 elements, otherwise the audit log entry will simply state *<changed>*.

The history record will contain:

| Table 2 History record description | |
| --- | --- |
| Field | Description |
| Parent entity  (entity key) | The database entity this audit trail history belongs to. This field contains the unique database identifier of this entity. |
| Action  (operation) | Possible values are:   * update * create * delete * rename * authorized |
| 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. |
| 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 field values.  It also includes a list of all the properties that have been deleted from the parent entity. |

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.

If the user decides to remove an entity, the entity is removed from the database and the entire audit trail log is transferred into the area for removed entities. This gives the possibility to reconstruct deleted entities if there is a need.

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
* Physical device type entities with two types of optional properties
* Physical devices with optional properties
* 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 39 and Figure 40.

##### 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 | {  "name" : "Device status",  "description" : "Describes a status of the physical device",  "scalar" : true,  "definition" : ["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 | {  "name" : "Device status",  "description" : "Describes a status of the physical device",  "scalar" : true,  "definition" : ["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 | NULL |

##### 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 | {  "name": "Ampere",  "quantity": "Electric current",  "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 | {  "name" : "Current",  "description" : "Electric current",  "association" : "TYPE",  "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 | {  "fields" : {  "name": "ControlLogix AC Power Supply",  "description" : "Rockwell"  },  "properties" : null,  "tags" : null,  "relationships" : null  } |

| 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 | {  "fields" : {  "name": "Power equipment",  "description" : "Power equipment logical group",  "is\_hosting\_slot" : false,  "is\_root" : false  },  "properties" : null,  "tags" : null,  "relationships" : {  "added" : [  {  "relation" : "includes",  "parent\_id" : 10,  "parent\_name" : "Inventory",  "child\_id" : 11,  "child\_name" : "Power equipment"  }  ],  "removed" : null  }  } |

| 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 | {  "fields" : {  "name": "Power equipment",  "description" : "Power equipment logical group",  "is\_hosting\_slot" : false,  "is\_root" : false  },  "properties" : null,  "tags" : null,  "relationships" : {  "added" : [  {  "relation" : "includes",  "parent\_id" : 11,  "parent\_name" : "Power equipment",  "child\_id" : 10345,  "child\_name" : "ISrc-01:PS-A01"  }  ],  "removed" : null  }  } |

| 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 | {  "fields" : {  "name": "ISrc-01:Chop-A01",  "description" : "Ion source chopper",  "component\_type" : "Chopper",  "is\_hosting\_slot" : true,  "beamline\_position" : 312.30,  "global\_x" : 91.44,  "global\_y" : 287.13,  "global\_z" : 1.0,  "global\_roll" : 0.0,  "global\_yaw" : 0.0,  "global\_pitch" : 0.0  },  "properties" : null,  "tags" : null,  "relationships" : {  "added" : [  {  "relation" : "includes",  "parent\_id" : 52,  "parent\_name" : "ISrc",  "child\_id" : 9182,  "child\_name" : "ISrc-01:Chop-A01"  }  ],  "removed" : null  }  } |

| 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 | {  "fields" : {  "name": "ControlLogix AC Power Supply",  "description" : "Rockwell"  },  "properties" : {  "added\_modified" : [  { "name": "Current", "value" : null }  ],  "deleted" : null  },  "tags" : null,  "relationships" : 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 | {  "fields" : {  "name": "ControlLogix AC Power Supply",  "description" : "Rockwell"  },  "properties" : {  "added\_modified" : [  { "name": "Weight", "value" : 132.42 }  ],  "deleted" : null  },  "tags" : null,  "relationships" : 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 | {  "fields" : {  "name": "ControlLogix AC Power Supply",  "description" : "Rockwell"  },  "properties" : {  "added\_modified" : null,  "deleted" : [ "Weight" ]  },  "tags" : null,  "relationships" : null  } |

##### Relationship removal

The relationship removal (usually used for removing the additional relationship from the installation slot) is recorded in the following way.

| Table 15 Relationship removal | |
| --- | --- |
| Name | Value |
| Parent entity | Unique identifier of the “Power equipment” container DB entity |
| Action | update |
| User | mvitorovic |
| Timestamp | 2014-05-21 10:42:42 |
| Entity type | Slot |
| Name | Includes |
| Modified properties | {  "fields" : {  "name": "Power equipment",  "description" : "Power equipment logical group",  "is\_hosting\_slot" : false,  "is\_root" : false  },  "properties" : null,  "tags" : null,  "relationships" : {  "added" : null,  "removed" : [  {  "relation" : "includes",  "parent\_id" : 11,  "parent\_name" : "Power equipment",  "child\_id" : 10345,  "child\_name" : "ISrc-01:PS-A01"  }  ]  }  } |

Note that in this case there could be multiple relationships, so the delete record has to specify exactly which relationship was removed. Also note, that when an installation slot is removed, the installation slot removal record is preceded by removal records for all relationships the device was in.

##### Working with tags

| Table 16 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).

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

To merge the two authorization models, the CCDB application will need to define RBAC permissions that are composed of the application resource name and the application permission. Examples:

* SLOT\_CREATE
* COMPTYPE\_DELETE
* DEVICE\_RENAME

## 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 a menu titled *Data definitions* which only a super user can see. Inside this menu the administrator can browse:

* device type definitions
* containers
* property definitions
* units
* enumeration definitions

#### Enumerations

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

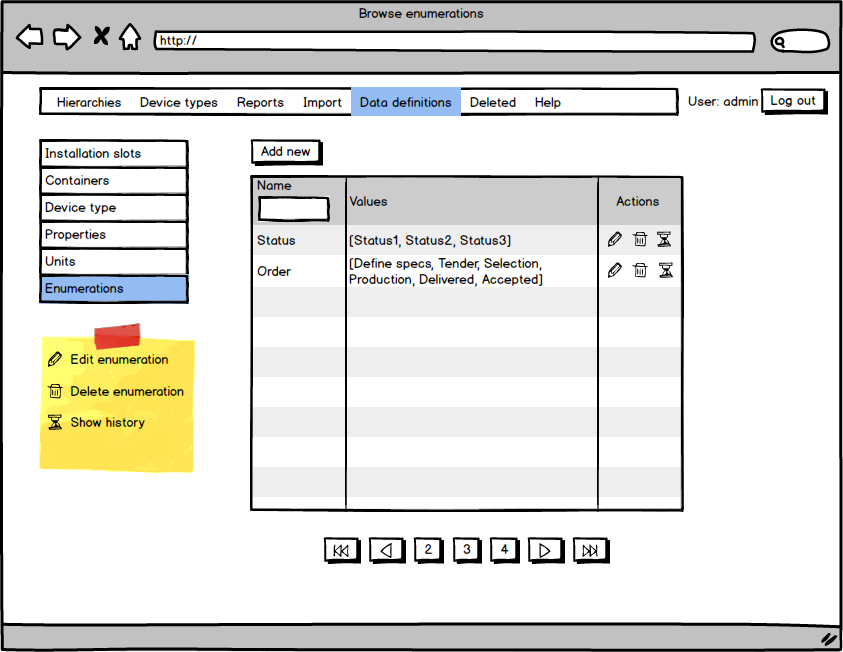


Figure 10 Browse enumerations screen

On this screen the super user can either add a new enumeration definition or edit an existing one.

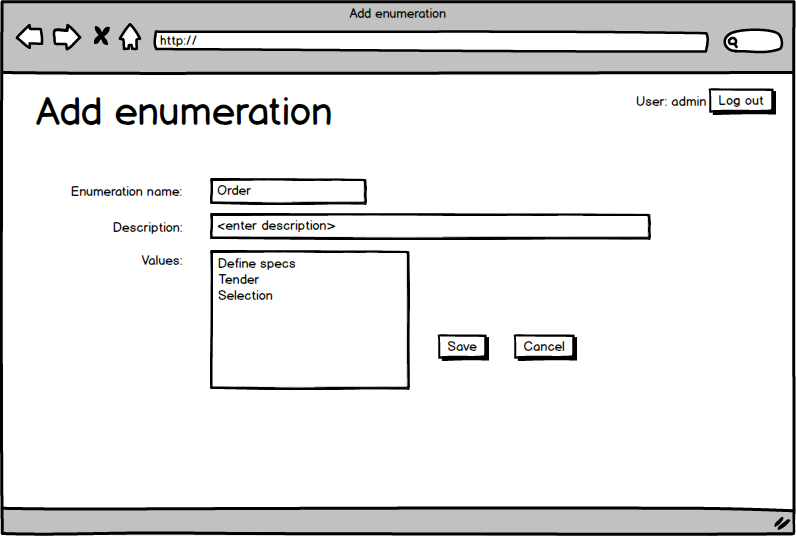


Figure 11 Defining enumeration

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.

See also section 3.1.6.2.

#### 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 Data definitions 🡪 Units menu.

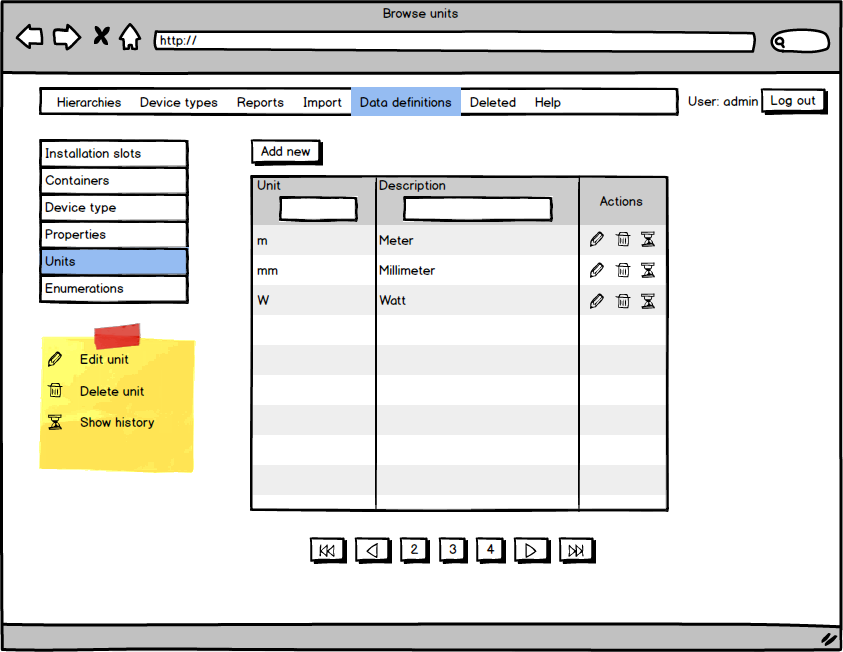


Figure 12 Unit definitions

Pressing Add new or the edit action brings the user to the unit definition screen.

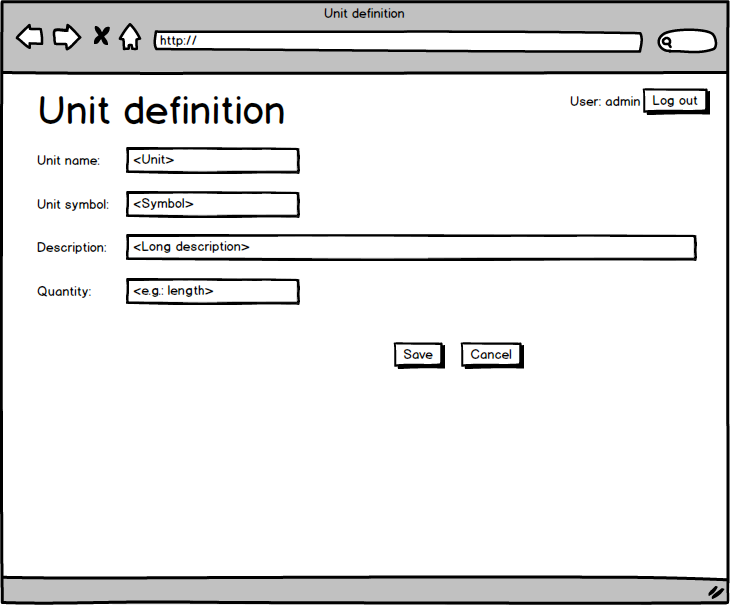


Figure 13 Unit definition

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.

See also section 3.1.6.1.

#### 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:

* Containers
* Installation slots
* Device types
* Device instances

Properties have a name, a description, a data type, and an optional unit (CCDB\_170). 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 Data definitions 🡪 Properties menu.

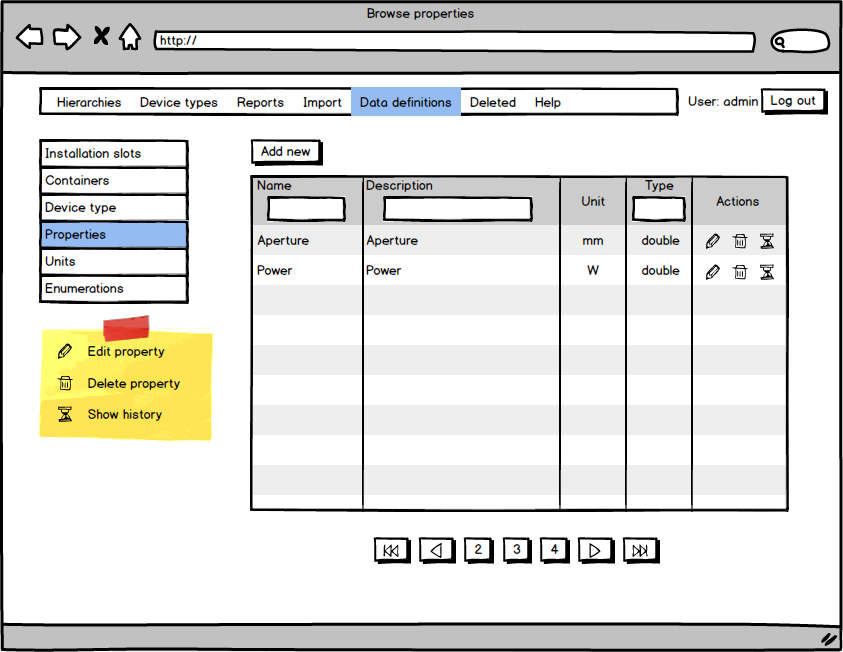


Figure 14 Browse properties screen

Pressing Add new or the edit action brings the user to property definition screen.

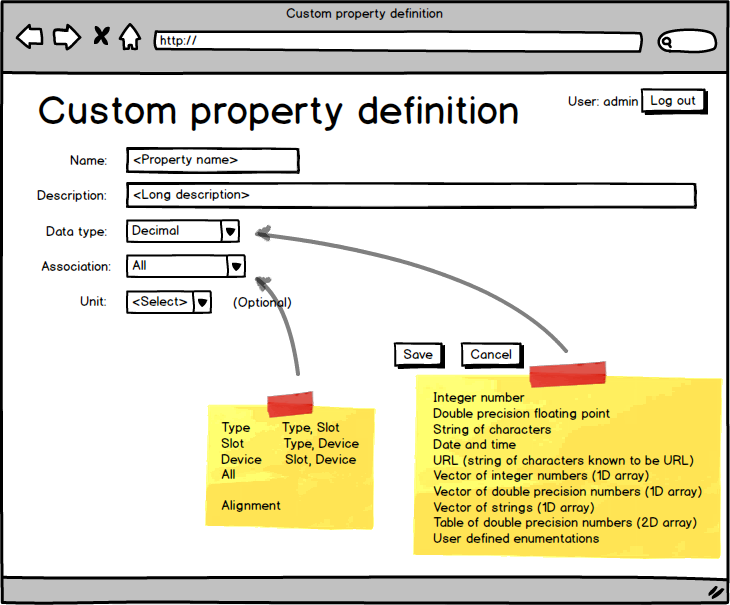


Figure 15 Property definition screen

#### 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 for which the device type was defined. The device types are defined by the super user through the Data definitions 🡪 Device type menu.

The screen offers to Add new or edit (CCDB\_010, CCDB\_020, CCDB\_025, CCDB\_027, CCDB\_028) an existing device type. The screen also contains an action for manipulation the properties associated with each device type. There are also two convenience actions that give the super user an option of viewing all devices of some type and adding a new device instance (CCDB\_010, CCDB\_020, CCDB\_025, CCDB\_027, CCDB\_028) of a certain type.

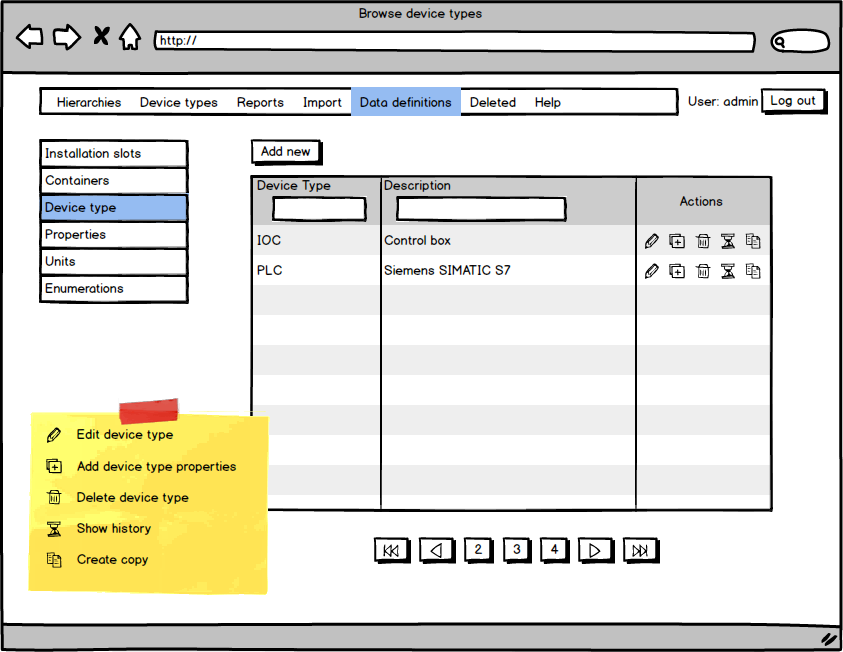


Figure 16 Device type listing

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 of this device defined in the CCDB application.

Listing all the devices of a specific type is also the most reliable way of locating an instance of a specific device. But it will be always accessible here.

Copying a device type copies the device type with all the properties and property definitions. The user needs to specify a new name.

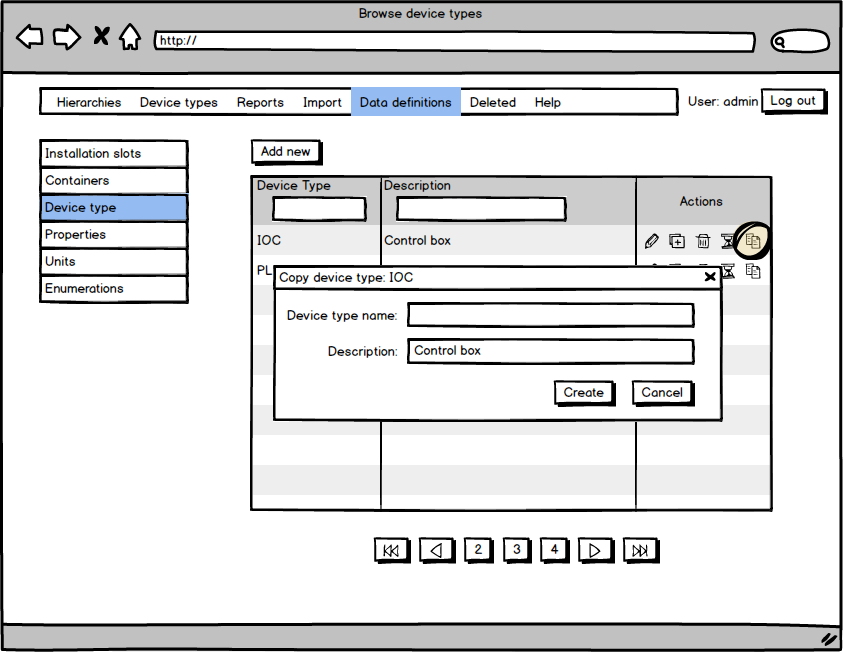


Figure 17 Copying of a device type

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

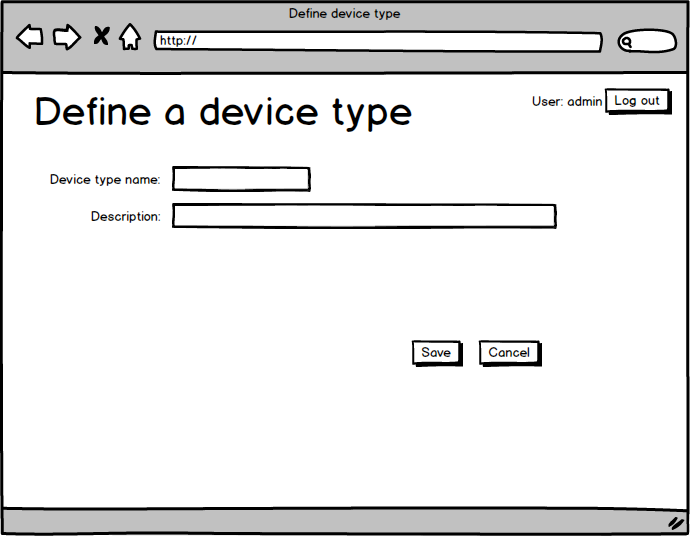


Figure 18 Define device type

Saving the device type or pressing Cancel takes the user to Device type listing screen.

#### 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 properties icon in the device type Actions column which opens the device type properties definition screen.

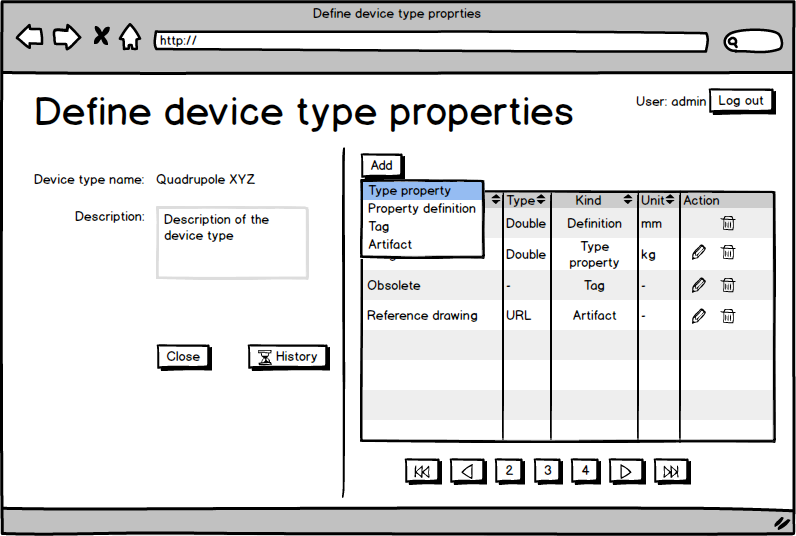


Figure 19 Defining device type properties

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. *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. *property definition*s  
   these are properties that hold different values for each device instance, but each device instance automatically has them and has to specify a value for them. Such properties also cannot be removed from the device instance. For property definition it is only possible to use properties which can be associated by the device instance.
3. *tags*see section 3.1.7.4
4. *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 are displayed at the device instance screen.

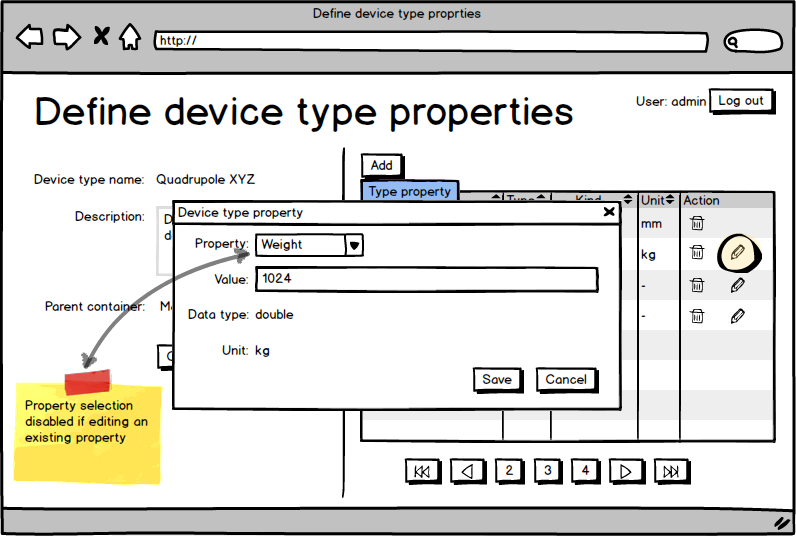


Figure 20 Modifying device type property

The device type properties are set only once by the CCDB super user meaning that authorized user cannot redefined such a property at the device type level. Super user can still modify the property.

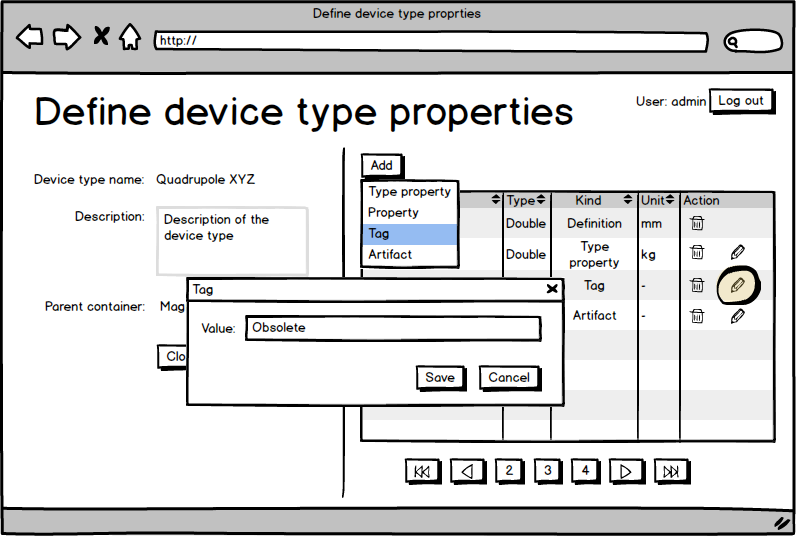


Figure 21 Adding or editing a tag

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).

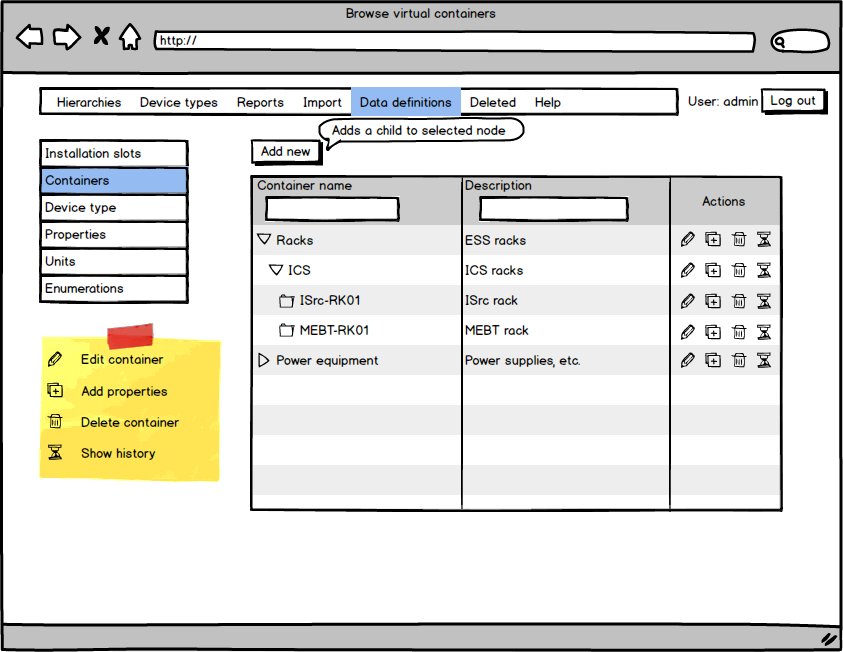


Figure 22 Browse containers

Container does not have many attributes besides name and an optional parent container (Figure 23) but additional properties can be assigned to it (Figure 24). A container without a parent is a root of some hierarchy.

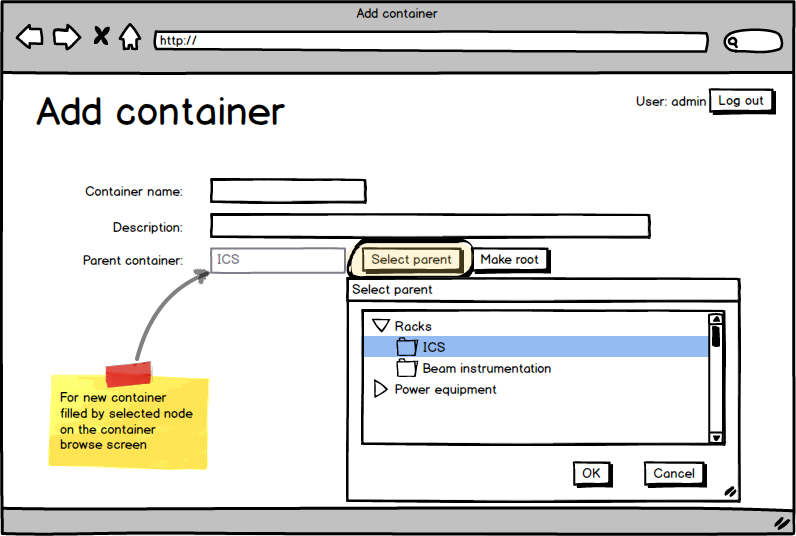


Figure 23 Define container

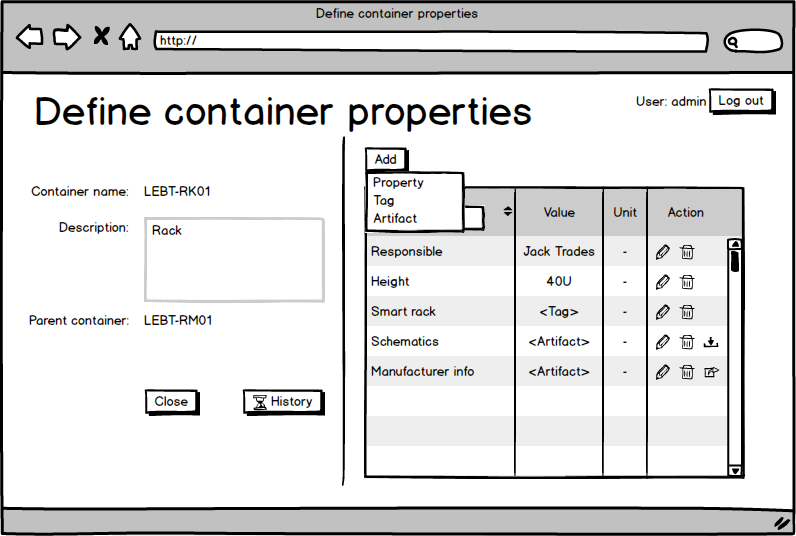


Figure 24 Container properties

Each container can have any properties and tags associated with it. They can only be added and edited by the super user, 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.

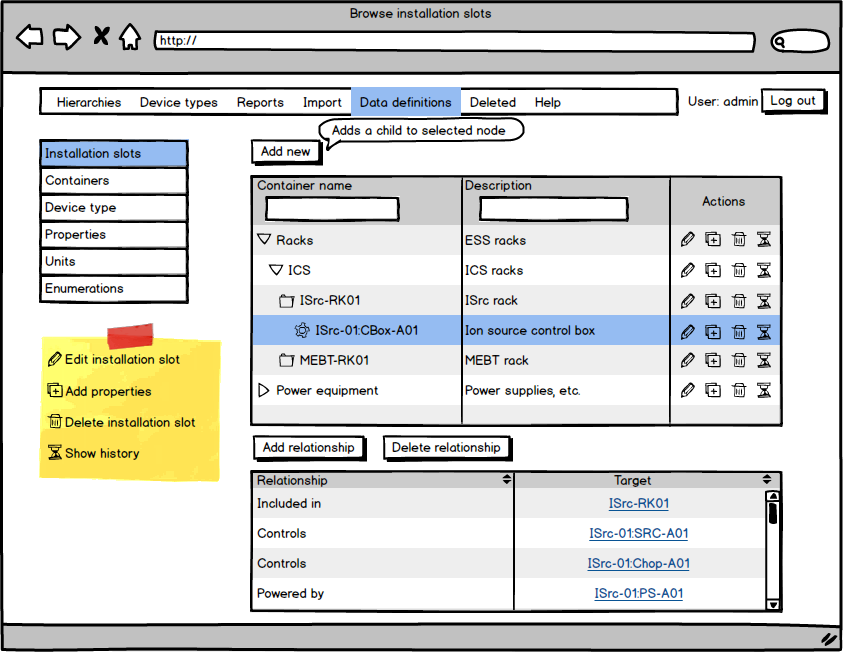


Figure 25 Browse installation slots

Pressing the Add new 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 other optional information, such as the description, the position and the global coordinates of the equipment.

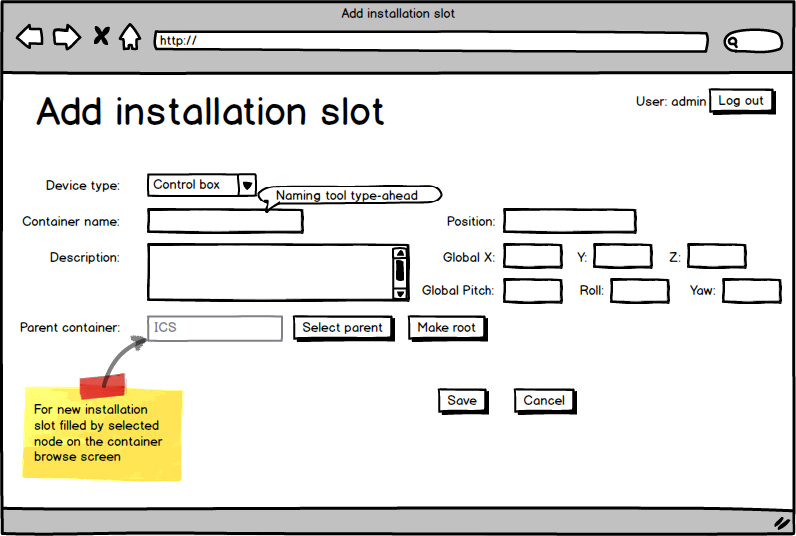


Figure 26 New installation slot

In case of the lattice device, the position and the global coordinates equal to the beamline position and the global coordinates of the device. If the device is not a lattice device, the position field can be used to specify the order of the installation slots within its parent.

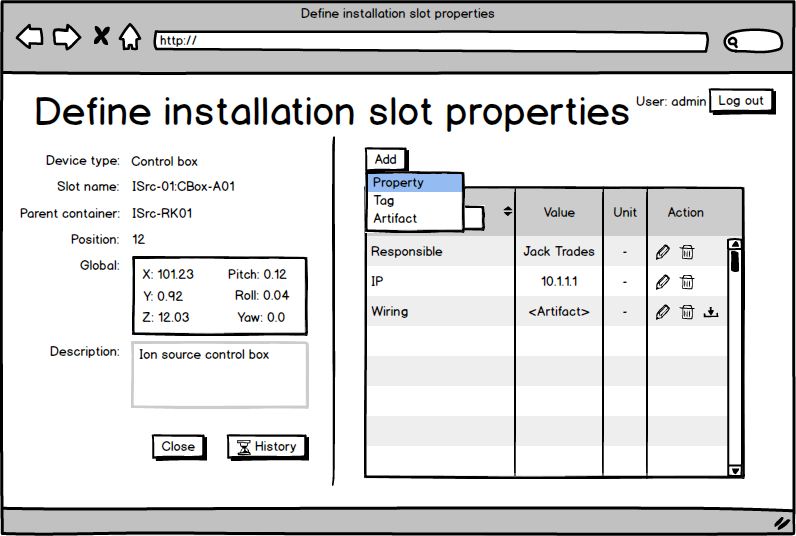


Figure 27 Installation slot properties

Changing and adding installation slot properties (CCDB\_106) is identical to container functionality (Figure 20).

The installation slots can be in relationship to other containers and installation slots in the database, which is manipulated in the *Browse installation slots* screen (Figure 25, Figure 28). 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).

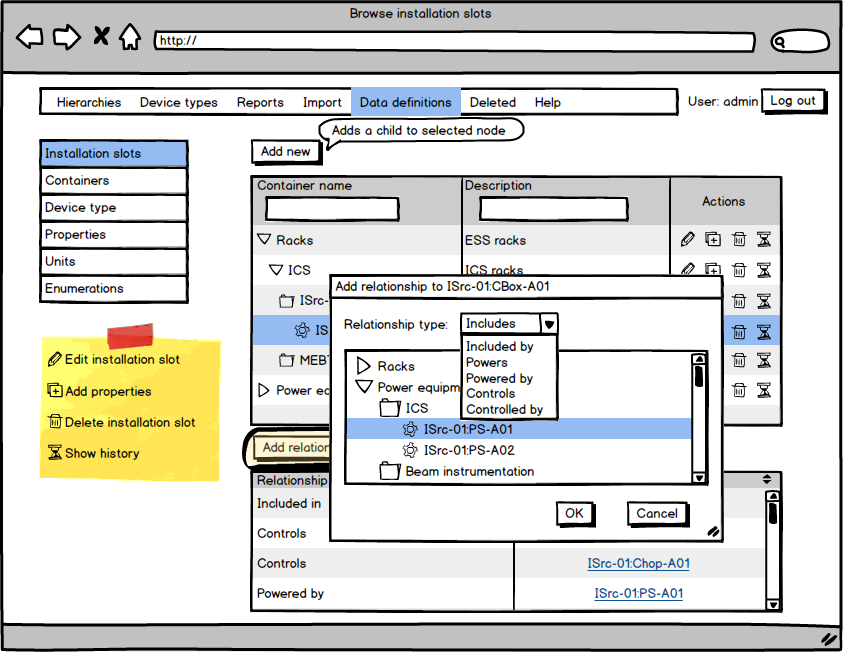


Figure 28 Adding relationships to the installation slots

### 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 device types, and selecting a device type gives you a list of all device instances of that type.

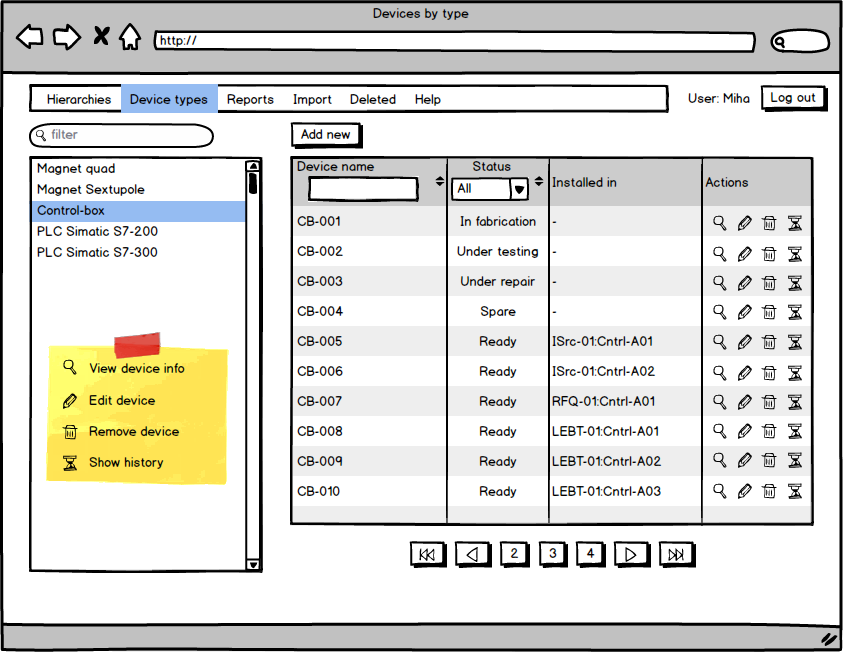


Figure 29 Listing device instances by device type

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 new button opens a screen for adding a new device instance to the database.

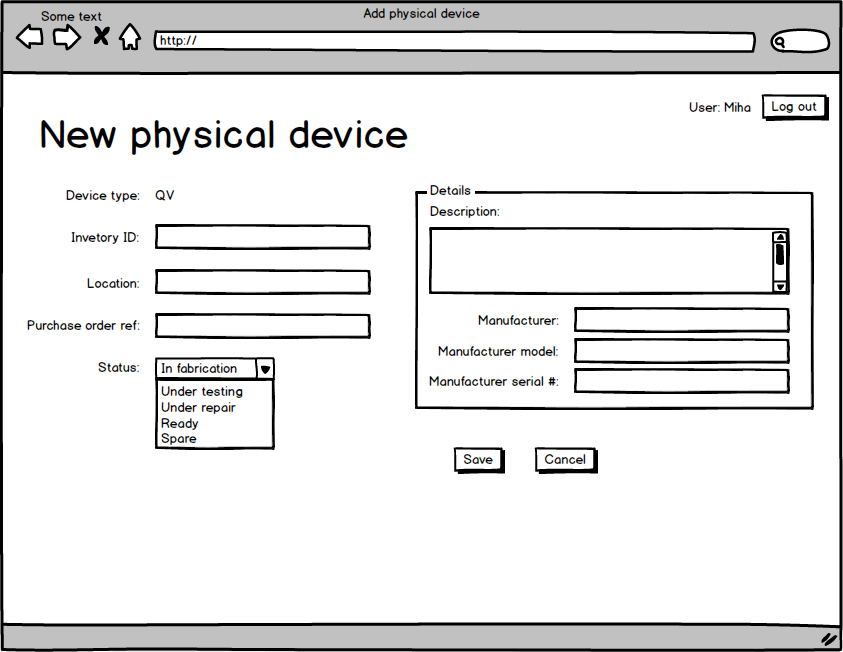


Figure 30 Adding a new physical device (CCDB\_060, CCDB\_070, CCDB\_073, CCDB\_110)

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

In the device details section, the device *manufacturer* information is mandatory, and the information on *manufacturer model* and *serial number* is optional. If the *manufacturer serial number* is provided, the information for *manufacturer* and *manufacturer model* must be provided as well. In this case the application requires that the triplet *manufacturer*, *manufacturer model* and *manufacturer serial number* is unique.

A device instance can either be installed in the control system or not (spare, test equipment, in service), which is indicated by the Status field and the installation record.

#### Device instance details

Pressing the device information icon displays the device details screen shown in Figure 31. The screen shows basic device information, which is similar for all device types.

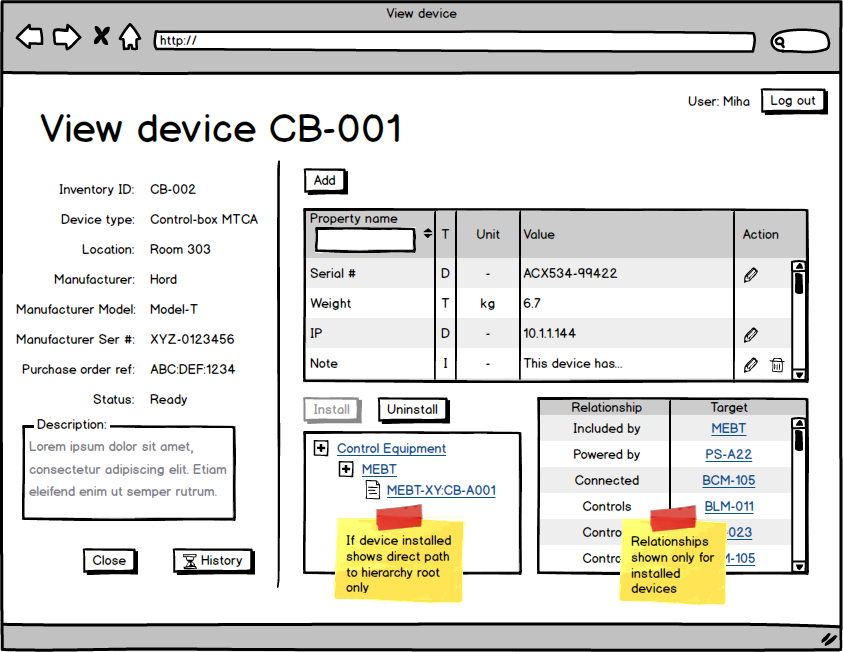


Figure 31 The device details screen

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

| Table 17 Device property types | |
| --- | --- |
| Property type | Description |
| T | Device type property; identical for all devices of the same type. Such properties cannot be edited or removed. |
| D | Device property; defined at the device type level, but different for each instance. Such properties cannot be removed. |
| I | Instance property; such properties have been appended to this device instance only and can be removed. |
| G | Device type tag; a tag associated with a device type. Such tags cannot be edited or removed. |
| A | Tag; tag is a simple text value associated with a device instance. Such tags can be removed. |

The table in the lower right corner displays all the CCDB instances this device is in relationship with. The table displays the type of relationship and the CCDB entity that the device instance is in relationship with. Clicking on the ID of the target entity opens that entity for inspection. The table also includes the links to the device this device connects to. This information is obtained from Cabling application.

Pressing the Add property button or the *edit* icon opens the property editing dialog show in Figure 32.

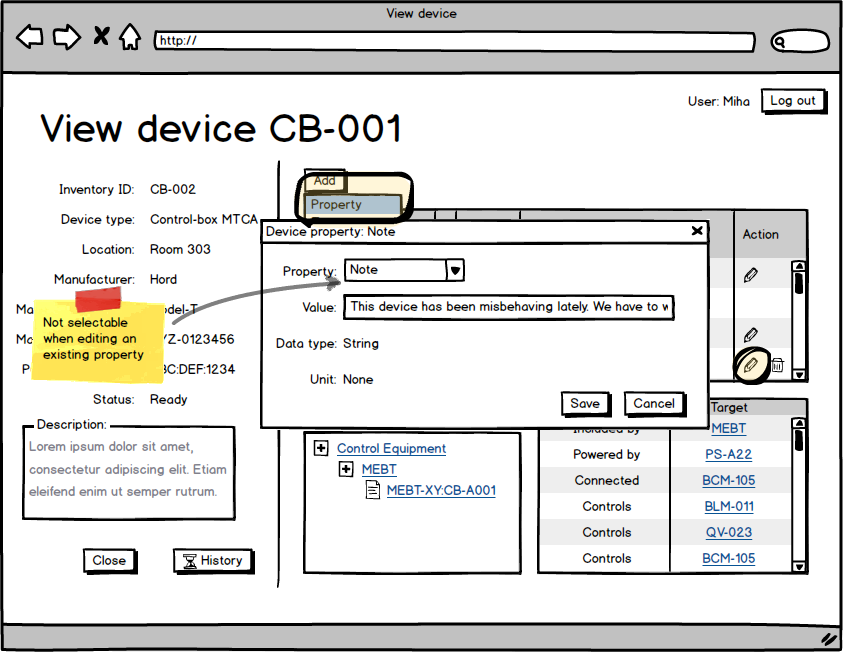


Figure 32 Editing device instance property

#### Device installation

Devices which are not installed yet can be installed into an installation slot by pressing on the *install* button.

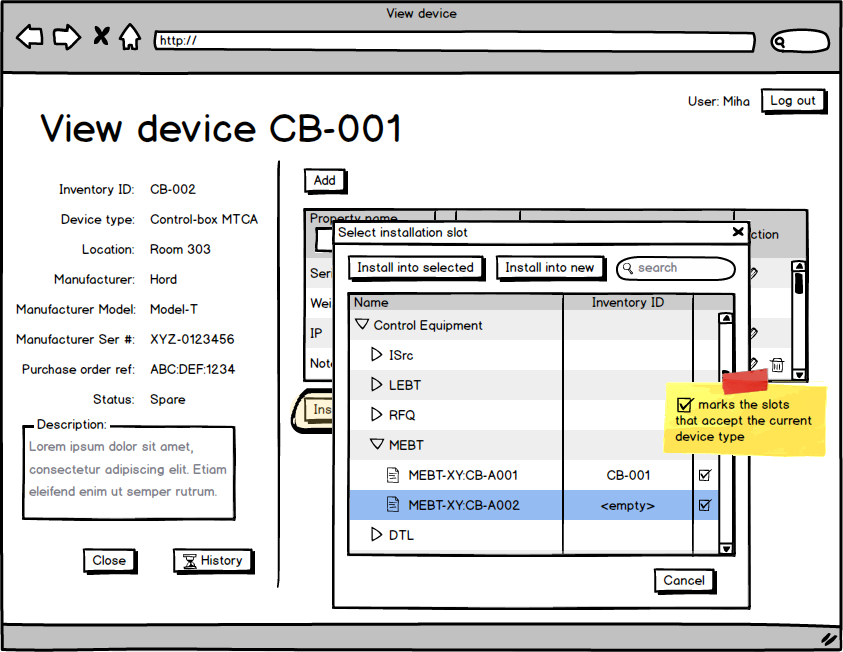


Figure 33 Device installation

This opens a dialog where the user can browse the configuration hierarchies (see section 3.3.2.5) and installed the device into an empty installation slot of appropriate type.

The user can also create a new installation slot of appropriate type as a child of a currently selected item.

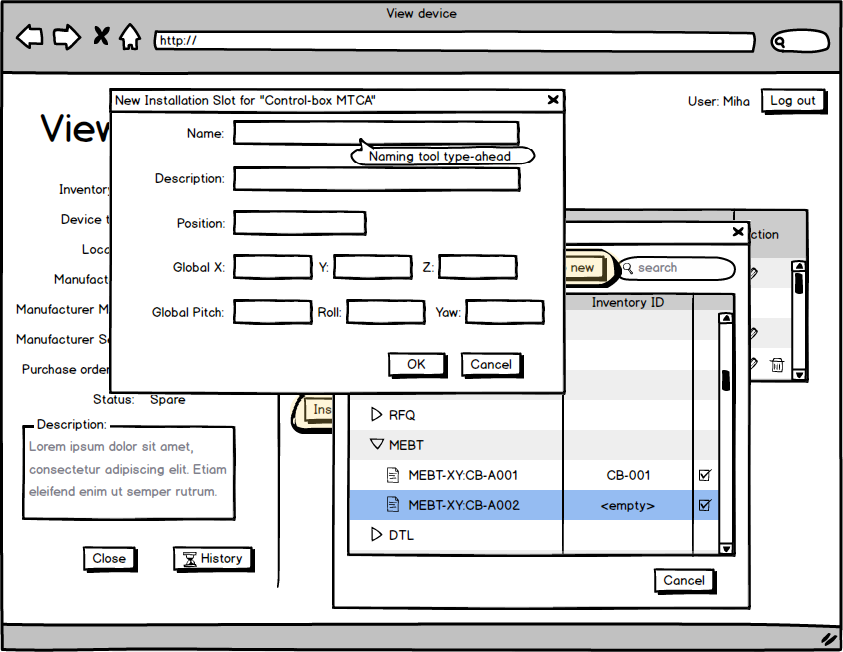


Figure 34 Install device into a new slot

The position and coordinate fields are double precision numbers and are optional if they do not apply (on the elements which are not on the lattice). The position field can still be used to specify the order of the elements in the container if this is desired. If the position is not specified for all elements, they will be listed in the chronological order.

#### CCDB hierarchies

The CCDB uses the *includes* relationship to construct various hierarchies of containers and devices. The user can browse those hierarchies by opening the Hierarchies menu.

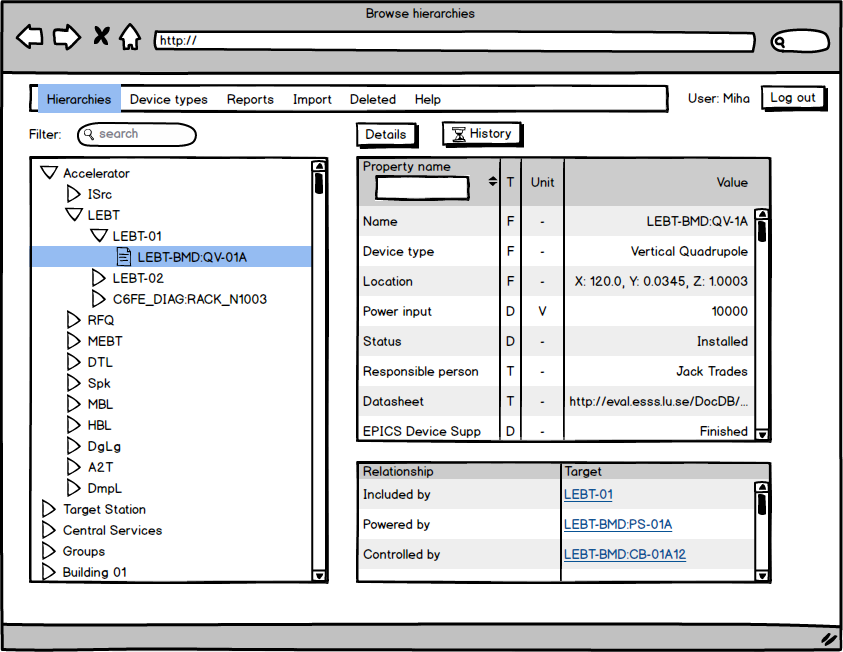


Figure 35 Browsing hierarchies - device

In this view the user can select an item from the hierarchical display control on the left and see an overview of the entity properties on the right. Clicking on the Details button opens a screen shown in Figure 31.

Similarly, choosing a container in the left hand view shows and overview of the selected container.

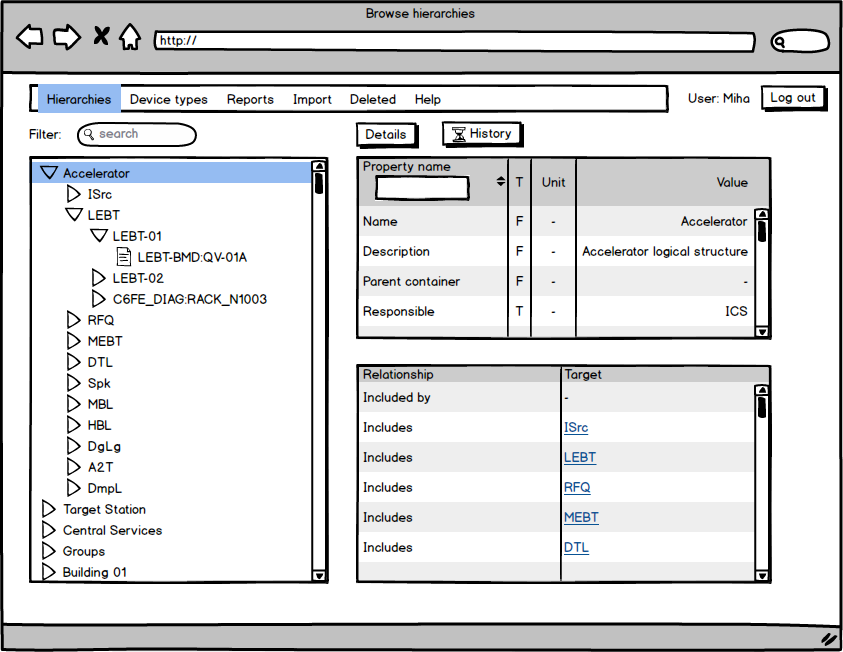


Figure 36 Browsing hierarchies - container

### 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) and CSV (tab separated value) 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.

#### Super user import

Upon selecting the type of data to import in the Import menu, the super user is shown a screen depicted in Figure 37. In the beginning the user sees File input control, a Browse button and an Upload button. The user can select the Excel spreadsheet file from his disk by clicking on the Browse button. The user can also download a sample worksheet file that he can fill with his own data by pressing the Download sample file button.

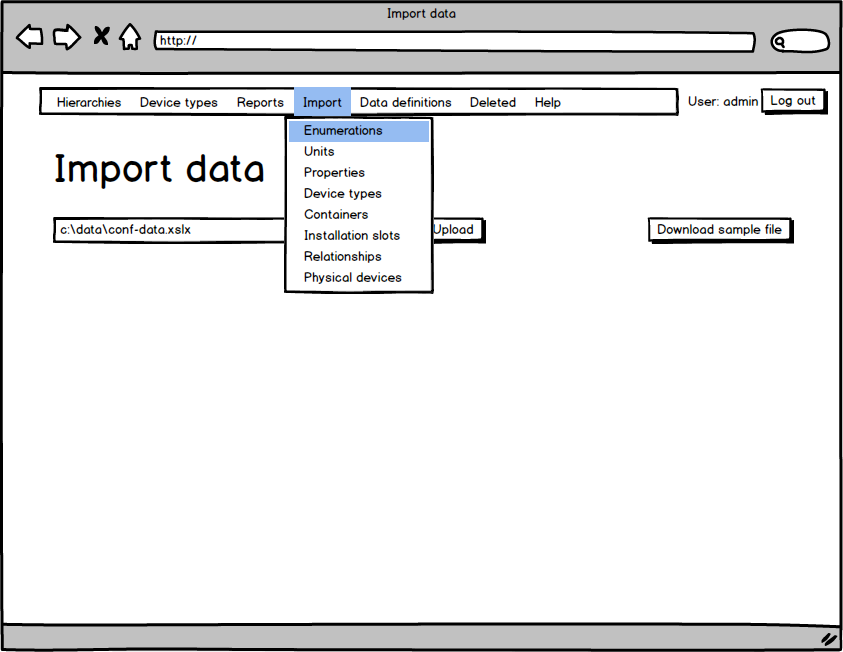


Figure 37 Data import screen

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 of the containers and installation slots import the user is able to select the second file. This file should contain the relationships in case the containers or slots file has any new entities. The relationships file should contain these new entities somewhere into the database hierarchy. If any container or installation slot is left orphaned the import fails and no data is imported. The import also fails if the user tries to assign a parent to the root node.

In case the containers or installation slots file contains only deletions or modifications to existing database entities, the relationships file is not required.

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

#### Authorized user import

Upon selecting the Import menu, the authorized user only sees the operations which are permitted by his application permissions. Otherwise the user interface is identical to the super user UI and is shown in Figure 37.

#### 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
* Relationships
* 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. Possible import commands are:

* Header
* Update
* Rename
* Delete
* End
* <no value>

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.

The *header* command instructs the CCDB application this line contains the column headers, that is fields and properties into which the appropriate data needs to be inserted. The contents of each non-empty cell must exactly match one of the fields in the imported data type or a defined property name (where properties can be used). If a cell is empty, the import procedure will skip this column. If the column contains a value that cannot be matched to a data record field or property, the error is reported and the import is aborted. The same worksheet can contain multiple *header* lines, which can redefine the columns being imported – this is very useful if the same worksheet contains instances of the same data type that have different properties associated with them. But in all cases the first data column must contain the identifying field of the data type.

The *update* command instructs the CCDB application to either create a new instance of the appropriate data record, or 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 import procedure continues without error. The information in all other columns is ignored and can be omitted.

The *rename* command instructs the application to rename the entity specified in the name column. The name column must contain information formatted in the following way: [old name] new name. For the rename command the information in all other columns is ignored and can also be omitted.

The *end* command instructs the CCDB application to stop the importing data on this worksheet in proceed to the next worksheet. Even if subsequent lines contain valid data they will not be imported.

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 38.

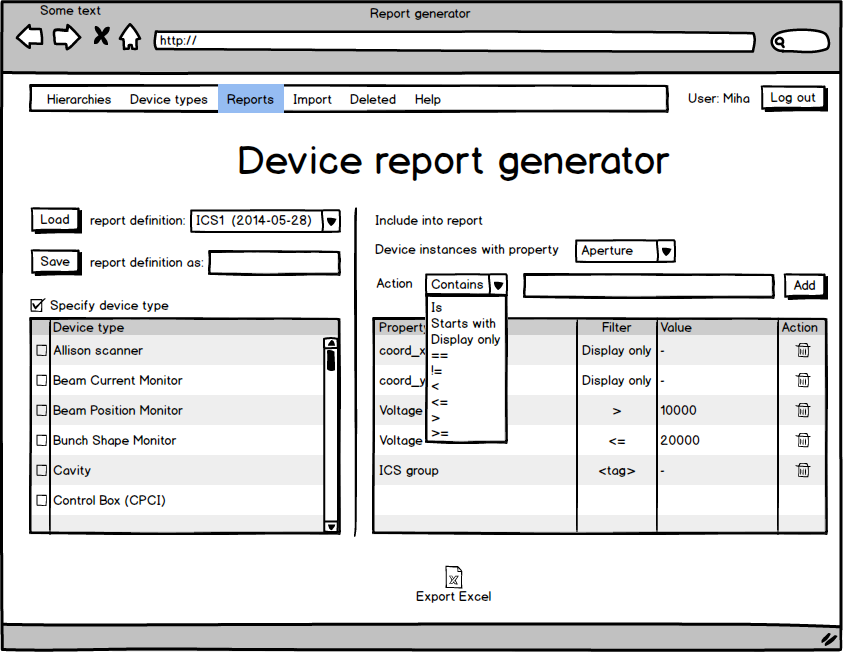


Figure 38 Report generation screen

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.

### 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. To access audit log entries for deleted entities the users can navigate to the deleted menu. Here they can access information on the deleted entries in the CCDB application.

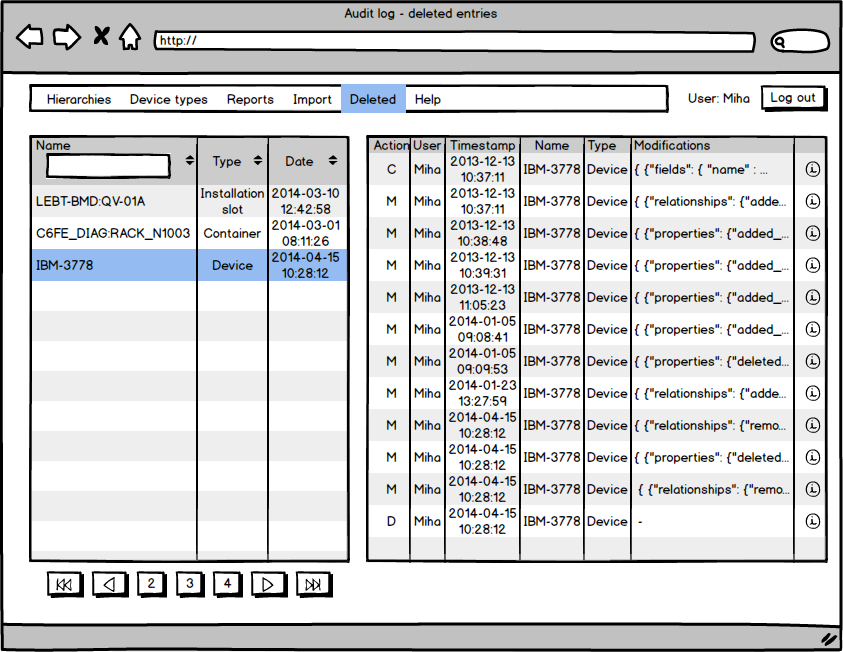


Figure 39 Audit log for deleted entities

On the left side the user sees a list of all deleted entities initially sorted by deletion 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 press the *information* icon for the desired log entry. Log entry details are shown in Figure 40.

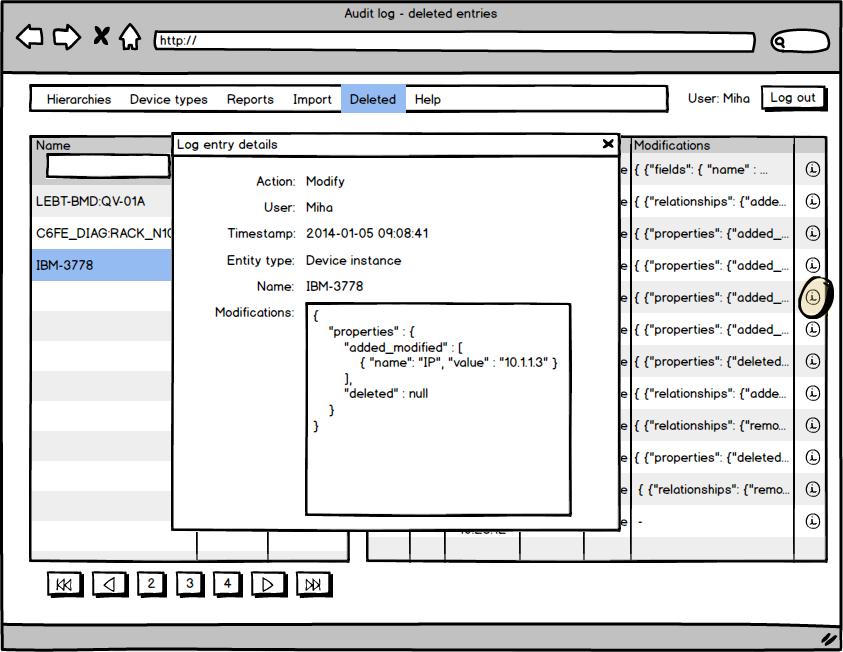


Figure 40 Audit log entry details

## 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, device instances and audit history logs.

#### Device types

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

* tag
* prop\_name
* prop\_value

Using these parameters, the requestor can retrieve a list of all the device types with a certain tag, a list of device types that have a certain property (value of the property is not important), or a list of device types having a certain property with the specified value. Specifying only a property value without a property name will return an error.

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

##### Examples

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

<collection>  
 <deviceType>  
 <name>CPU</name>  
 <description>CPU</description>  
 </deviceType>  
 <deviceType>  
 <name>CPU\_1</name>  
 <description>CPU</description>  
 </deviceType>  
 <deviceType>  
 <name>CRATE</name>  
 <description>Crate</description>  
 </deviceType>  
 ...  
</collection>

Figure 41 Sample RESTful service output listing all device type

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

<deviceType>  
 <name>CPU</name>  
 <description>CPU</description>  
</deviceType>

Figure 42 Sample RESTful service output showing a specific device type

#### Installation slots

The RESTful interface under the URL <http://server/rest/slot> will return a list of all the installation slots in the CCDB application. The URL will support the following parameters:

* deviceType
* tag
* prop\_name
* prop\_value

The service will return a list the installation slots:

* all installation slots of specified device type,
* all installation slots being tagged with a specific value,
* all installation slots that have a specified property (the value of the property is not important),
* all installation slots having a certain property with the specified value.

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

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

##### Examples

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

<collection>  
 <installationSlot>  
 <name>H1:3:1:DC07</name>  
 <desription>DIO</desription>  
 <deviceType>  
 <name>DigitalCard01</name>  
 <description>Digital Card 01</description>  
 </deviceType>  
 <parents/>  
 <children/>  
 <powers/>  
 <poweredBy>  
 <powerBy>H1:2:1:DC06</powerBy>  
 </poweredBy>  
 <controls/>  
 <controlledBy/>  
 <properties>  
 <propertyValue>  
 <name>TypeProp</name>  
 <value>-1.1</value>  
 <dataType>Double</dataType>  
 </propertyValue>  
 <propertyValue>  
 <name>SignalName1</name>  
 <value>null</value>  
 <dataType>String</dataType>  
 </propertyValue>  
 <propertyValue>  
 <name>SignalName2</name>  
 <value>null</value>  
 <dataType>String</dataType>  
 </propertyValue>  
 </properties>  
 </installationSlot>  
 ...  
</collection>

Figure 43 Sample RESTful service output listing all installation slots

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

<collection>  
 <installationSlot>  
 <name>CPU1</name>  
 <desription>Hello</desription>  
 <deviceType>  
 <name>CPU</name>  
 <description>CPU</description>  
 </deviceType>  
 <parents>  
 <parent>LEBT-00:Ctrl-Crate-1</parent>  
 </parents>  
 <children/>  
 <powers/>  
 <poweredBy/>  
 <controls/>  
 <controlledBy/>  
 <properties>  
 <propertyValue>  
 <name>Manufacturer</name>  
 <value>Intel</value>  
 <dataType>String</dataType>  
 </propertyValue>  
 <propertyValue>  
 <name>Model</name>  
 <value>i7</value>  
 <dataType>String</dataType>  
 </propertyValue>  
 </properties>  
 </installationSlot>  
 ...  
</collection>

Figure 44 Sample RESTful service output listing all installation slots of specified device type

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

<installationSlot>  
 <name>CPU1</name>  
 <desription>Hello</desription>  
 <deviceType>  
 <name>CPU</name>  
 <description>CPU</description>  
 </deviceType>  
 <parents>  
 <parent>LEBT-00:Ctrl-Crate-1</parent>  
 </parents>  
 <children/>  
 <powers/>  
 <poweredBy/>  
 <controls/>  
 <controlledBy/>  
 <properties>  
 <propertyValue>  
 <name>Manufacturer</name>  
 <value>Intel</value>  
 <dataType>String</dataType>  
 </propertyValue>  
 <propertyValue>  
 <name>Model</name>  
 <value>i7</value>  
 <dataType>String</dataType>  
 </propertyValue>  
 </properties>  
</installationSlot>

Figure 45 Sample RESTful service output showing a specific installation slot

#### 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/slotName>

<slotNames>  
 <name>H1:3:1:DC07</name>  
 <name>H1:1:1:DC02</name>  
 <name>H1:1:2</name>  
 <name>H1:3:2:DC08</name>  
 <name>H1:3:3:DC09</name>  
 <name>H1:1:2:DC03</name>  
 <name>H1:1:1:DC02:B01</name>  
 <name>H1:1:1:DC02:B02</name>  
 <name>DEMO</name>  
 <name>H1:2:1:DC05</name>  
 <name>H1:2:1:DC06</name>  
 <name>H1:2:2:DC07</name>  
 <name>H1:2:2:DC08</name>  
 <name>WAREHOUSE</name>  
 <name>GALLERY</name>  
 ...  
</slotNames>

Figure 46 Sample RESTful service output listing all installation slot and container names

URL: <http://server/rest/slotName?type=DigitalCard01>

<slotNames>  
 <name>H1:3:1:DC07</name>  
 <name>H1:1:1:DC02</name>  
 <name>H1:3:2:DC08</name>  
 <name>H1:3:3:DC09</name>  
 <name>H1:1:2:DC03</name>  
 <name>H1:2:1:DC05</name>  
 <name>H1:2:1:DC06</name>  
 <name>H1:2:2:DC07</name>  
 <name>H1:2:2:DC08</name>  
 <name>H1:1:1:DC02\_1</name>  
 <name>Cnt1:DIO:CARD0001\_1</name>  
 <name>H1:1:1:DC02\_2</name>  
 <name>Cnt1:DIO:CARD0001</name>  
</slotNames>

Figure 47 Sample RESTful service output listing installation slots of a specified device type

#### Device instances

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

* type
* installation\_slot
* tag
* prop\_name
* prop\_value

The service will return a list of device instances. The requestor can specify the device instances of a specific type, device instances that are included in a given container. The user can also obtain a list of devices that have a certain tag, have a specified property (the value of the property is not important), or have a certain property with the specified value.

The URL [http://server/ccdb/rs/devices/*<name>*](http://server/ccdb/rs/devices/%3cname%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.

# Requirements traceability

| Table 18 Requirements traceability | |  |
| --- | --- | --- |
| Number | Title | Realization (page) |
| CCDB\_010 | Control Boxes | 12, 39, 39, 40, 52 |
| CCDB\_020 | Machine Protection System (MPS) Devices | 12, 39, 39, 40, 52 |
| CCDB\_025 | Personal Protection System (PPS) Devices | 12, 39, 39, 40, 52 |
| CCDB\_027 | Infrastructure | 12, 39, 39, 40, 52 |
| CCDB\_028 | Other Devices | 12, 39, 39, 40, 52 |
| CCDB\_029 | Web-based User Interface for Add, Modify, and Delete | 33 |
| CCDB\_030 | Unique ID | 11 |
| CCDB\_040 | Naming Convention | 17, 53 |
| CCDB\_041 | Naming Convention Integration | 53 |
| CCDB\_042 | Naming Convention Integration | 53 |
| CCDB\_050 | Name | 16, 17 |
| CCDB\_060 | Serial Number | 53 |
| CCDB\_070 | Global Coordinates | 53 |
| CCDB\_073 | Orientation Angle | 53 |
| CCDB\_080 | Physical Size | 14, 20 |
| CCDB\_085 | Shape Description | 54 |
| CCDB\_090 | Rack Location | 14, 20 |
| CCDB\_103 | Cable Connection | 50 |
| CCDB\_106 | Water Connection | 14, 20, 37, 43, 50 |
| CCDB\_107 | Firmware and FPGA | 14, 19, 20, 54 |
| CCDB\_110 | Tracking Status | 14, 14, 20, 37, 53, 54 |
| CCDB\_111 | Status type | 14, 20, 37, 54 |
| CCDB\_140 | History | 21, 21, 21, 21 |
| CCDB\_150 | Relationship To Other Devices | 12, 16, 45 |
| CCDB\_160 | Documentation | 43 |
| CCDB\_170 | Name Value Pairs | 14, 37 |
| CCDB\_175 | Access Control | 9, 32 |
| CCDB\_180 | Excel Export | 63 |
| CCDB\_190 | Reports | 33, 62 |
| CCDB\_194 | RESTful Interface | 66 |
| CCDB\_195 | API - Java | 9 |
| CCDB\_197 | Web-based User Interface for Reports | 33, 62 |
| CCDB\_198 | Import Format | 59, 61 |
| 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/