Fields

Introduction

Open-AudIT can store information in additional fields that are associated with each device.

Once an Additional Field has been created it can be used in queries and groups just like any other attribute in the database.

How Does it Work?

Fields attributes are:

- Name: The name you are giving to this field. Ideally it should be unique.
- Type: This can be either a varchar (a text field) or a List (a list of values that can be selected).
- Placement: Where on the Device Details screen you would like to see this field appear.
- Group: The group of devices this field applies to. To apply it to all devices, use the All Devices group.

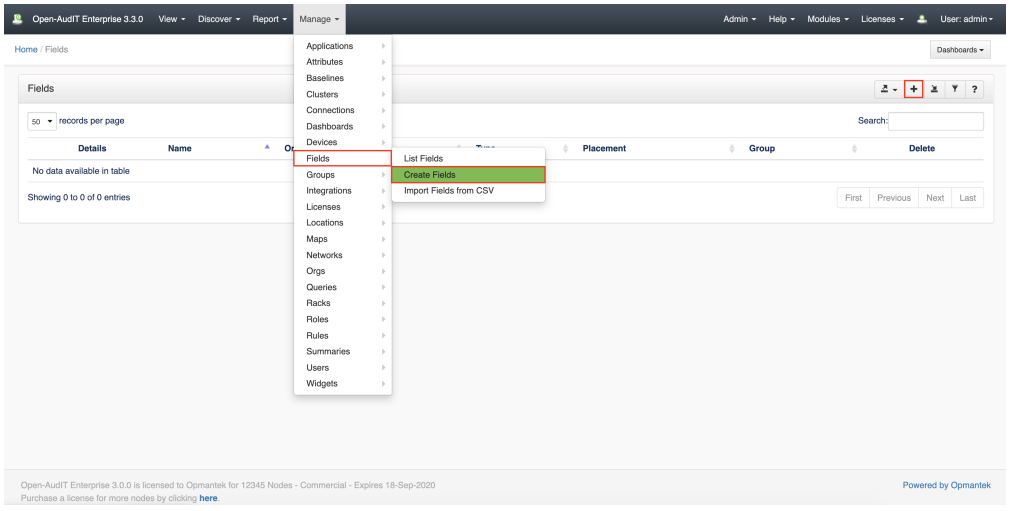
If you select a List type, an additional field will appear called Values. You should put a comma separated list of selectable values here. These will display in a dropdown box whenever the field is edited.

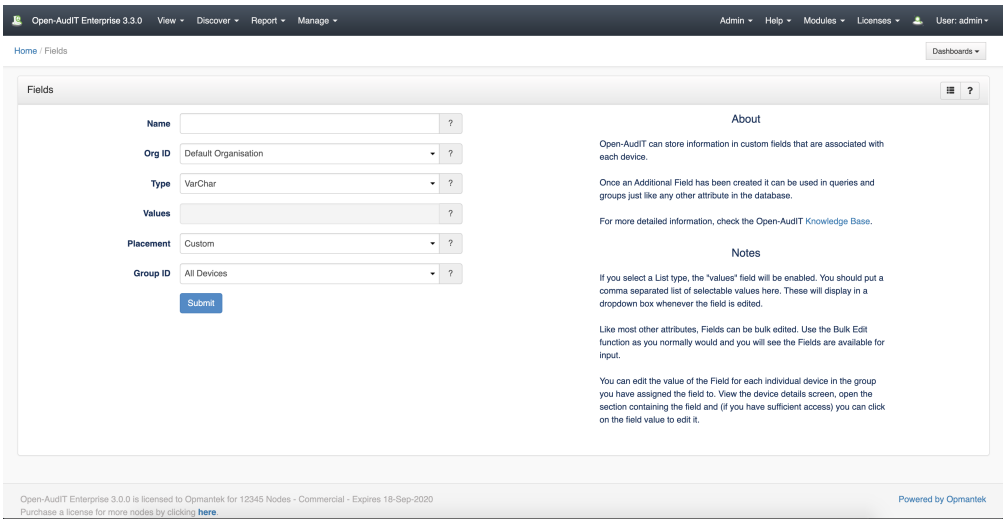
Like most other attributes, Additional Fields can be bulk edited. Use the Bulk Edit function as you normally would and you will see the Additional Fields are available for input.

You can edit the value of the Additional Field for each individual device in the group you have assigned the field to. View the device details screen, open the section containing the field and (if you have sufficient access) you can click on the field value to edit it, as below. In the below screenshot you can see I have assigned the "Test List" field to the 'custom' section.

Creating a Field

To make another network go to menu: Manage -> Fields -> Create Field.

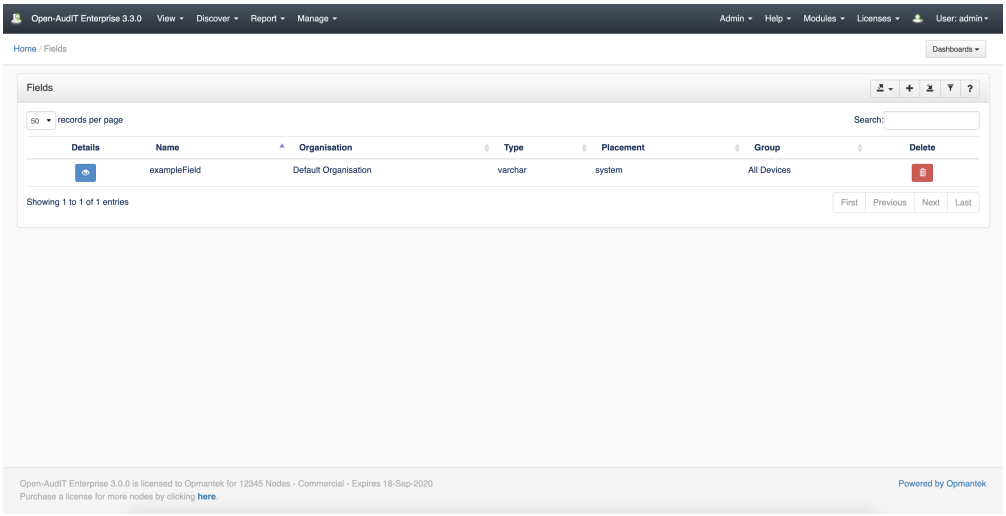




View Field Details

Go to menu: Manage -> Fields -> List Fields.

You will see a list of fields. You can view a field by clicking on the blue view icon. You can also edit or delete your fields.



Database Schema

The schema for the database is below. It can also be found in the application if the user has database::read permission by going to menu: Admin -> Database -> List Tables, then clicking on the "fields" table.

```
CREATE TABLE `fields` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(200) NOT NULL DEFAULT '',
  `org_id` int(10) unsigned NOT NULL DEFAULT '1',
  `group_id` int(10) unsigned NOT NULL DEFAULT '1',
  `type` enum('varchar','list') NOT NULL DEFAULT 'varchar',
  `values` text NOT NULL,
  `placement` enum('custom','system') NOT NULL DEFAULT 'system',
  `edited_by` varchar(200) NOT NULL DEFAULT '',
  `edited_date` datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```

Example Database Entry

Field are stored in the database in the "fields" table. A typical entry will look as below.

```
        id: 1
      name: Ink Type
    org_id: 2
  group_id: 8
      type: varchar
    values:
 placement: custom
 edited_by: Administrator
edited_date: 2017-05-22 00:26:38
```

API / Web Access?

You can access the /fields collection using the normal Open-AudIT JSON based API. Just like any other collection. Please see the API documentation for further details.

API Routes

Request Method	ID	Action	Resulting Function	URL Example	Notes	Example Response
GET	n		collection	/fields	Returns a list of fields.	fields_collection.json
GET	y		read	/fields/{id}	Returns a field's details.	fields_read.json
PATCH	y		update	/fields/{id}	Update an attribute of a fields entry.	fields_patch.json
POST	n		create	/fields	Insert a new fields entry.	fields_create.json
DELETE	y		delete	/fields/{id}	Delete a fields entry.	fields_delete.json

Web Application Routes

Request Method	ID	Action	Resulting Function	URL Example	Notes
GET	n	create	create_form	/fields/create	Displays a standard web form for submission to POST /fields.
GET	y	update	update_form	/fields/{id} /update	Show the connection's details with the option to update attributes using PATCH to /fields/{id}