Widgets

Widgets are designed to be used on Dashboards.

Enterprise users have complete freedom to create, update and delete widgets as required.

Creating

When creating a widget, if you select the 'Advanced' button you will have the ability to upload your own custom SQL.

The SELECT section of your SQL must contain fully qualified columns and specify certain columns.

Pie charts must return name or my_name, description or my_description and count.

Line charts must return date or my_date, description or my_description and count.

The WHERE section of your SQL must contain **`WHERE @filter`** so Open-AudIT knows to restrict your query to the appropriate Orgs.

SQL not containing this condition will result in the widget failing to be executed.

The SQL cannot contain **`WHERE @filter OR`**. That SQL will not be executed, however **`WHERE @filter AND`** queries are allowed.

An example widget SQL showing devices counted per location.

```
SELECT locations.name as `name`,
    locations.id AS `description`,
    count(system.id) AS `count`
FROM locations LEFT JOIN system ON (locations.id = system.location_id)
WHERE @filter
GROUP BY locations.name
```

More elaborate SQL can used to group attributes within a range. Below shows the SQL for devices not seen in a pie chart grouped by last seen date ranges.

```
SELECT IF ( system.last_seen = "2000-01-01", "unknown", ( IF ( system.last_seen < DATE(NOW() - INTERVAL 180
day), "180 Days or more",
( IF ( system.last_seen < DATE(NOW() - INTERVAL 150 day), "150-180 days", ( IF ( system.last_seen < DATE(NOW()
- INTERVAL 120 day), "120-150 days",
( IF ( system.last_seen < DATE(NOW() - INTERVAL 90 day), "90-120 days", ( IF ( system.last_seen < DATE(NOW() -
INTERVAL 60 day), "60-90 days",
( IF ( system.last_seen < DATE(NOW() - INTERVAL 30 day), "30-60 days", "7-30 days" ) ) ) ) ) ) ) ) ) ) ) ) AS
`my_name`,
IF ( system.last_seen = "2000-01-01", "system.last_seen=",
( IF ( system.last_seen < DATE(NOW() - INTERVAL 180 day), CONCAT("system.last_seen=<", DATE(NOW() - INTERVAL
180 day)),
( IF ( system.last_seen < DATE(NOW() - INTERVAL 150 day), CONCAT("system.last_seen=>", DATE(NOW() - INTERVAL
180 day), "&system.last_seen=<", DATE(NOW() - INTERVAL 150 day)),
( IF ( system.last_seen < DATE(NOW() - INTERVAL 120 day), CONCAT("system.last_seen=>", DATE(NOW() - INTERVAL
150 day), "&system.last_seen=<", DATE(NOW() - INTERVAL 120 day)),
( IF ( system.last_seen < DATE(NOW() - INTERVAL 90 day), CONCAT("system.last_seen=>", DATE(NOW() - INTERVAL 120
day), "&system.last_seen=<", DATE(NOW() - INTERVAL 90 day)),
( IF ( system.last_seen < DATE(NOW() - INTERVAL 60 day), CONCAT("system.last_seen=>", DATE(NOW() - INTERVAL 90
day), "&system.last_seen=<", DATE(NOW() - INTERVAL 60 day)),
( IF ( system.last_seen < DATE(NOW() - INTERVAL 30 day), CONCAT("system.last_seen=>", DATE(NOW() - INTERVAL 60
day), "&system.last_seen=<", DATE(NOW() - INTERVAL 30 day)),
CONCAT("system.last_seen=>", DATE(NOW() - INTERVAL 30 day), "&system.last_seen=<", DATE(NOW() - INTERVAL 7
day))) ) ) ) ) ) ) ) ) ) ) AS `my_description`,
count(system.id) AS `count`
FROM system
WHERE @filter AND DATE(system.last_seen) < DATE(NOW() - INTERVAL 7 day)
GROUP BY `my_name` ORDER BY system.last_seen;
```

Database Schema

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

```
CREATE TABLE `widgets` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL DEFAULT '',
  `org_id` int(10) unsigned NOT NULL DEFAULT '0',
  `description` text NOT NULL,
  `type` enum('line','pie','') DEFAULT 'line',
  `table` varchar(50) NOT NULL DEFAULT '',
  `primary` varchar(50) NOT NULL DEFAULT '',
  `secondary` varchar(50) NOT NULL DEFAULT '',
  `ternary` varchar(50) NOT NULL DEFAULT '',
  `dataset_title` varchar(200) NOT NULL DEFAULT '',
  `group_by` varchar(50) NOT NULL DEFAULT '',
  `where` text NOT NULL,
  `limit` smallint signed NOT NULL DEFAULT '0',
  `options` text NOT NULL,
  `sql` text NOT NULL,
  `link` text NOT NULL,
  `edited_by` varchar(200) NOT NULL DEFAULT '',
  `edited_date` datetime NOT NULL DEFAULT '2000-01-01 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

API / Web Access

You can access the /widgets collection using the normal Open-AudIT JSON based API. Just like any other collection (assuming your an Enterprise user). Please see the API documentation for further details.

Access is provided as part of a roles permissions. Widgets is a standard resource and can have create, read, update, delete and execute permissions.

The API routes below are usable from both a JSON Restful API and the web interface. The Web application routes are specifically designed to be called from the web interface (a browser).

API Routes

Request Method	ID	Action	Resulting Function	Permission Required	URL Example	Notes	Example Response
GET	n		collection	widgets::read	/widgets	Returns a list of widgets.	
POST	n		create	widgets::create	/widgets	Insert a new widgets entry.	
GET	y		read	widgets::read	/widgets/{id}	Returns a widgets details.	
PATCH	y		update	widgets::update	/widgets/{id}	Update the details of a widgets entry.	
DELETE	y		delete	widgets::delete	/widgets/{id}	Delete a widgets entry.	
EXECUTE	y	execute	execute	widgets::read	/widgets/{id} /execute	Execute a widget and return the resulting dataset.	

Web Application Routes

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