NodeConductor SugarCRM Documentation

Release 0.5.0

OpenNode

December 05, 2016

Contents

SugarCRM service provides an interface to SugarCRM system. It creates separate VM for each SugarCRM installation via NodeConductor OpenStack endpoints.

CHAPTER 1

Guide

1.1 Installation

- Install NodeConductor
- Clone NodeConductor SugarCRM repository

```
git clone git@code.opennodecloud.com:nodeconductor/nodeconductor-sugarcrm.git
```

- Install NodeConductor SugarCRM into NodeConductor virtual environment

```
cd /path/to/sugarcrm/
python setup.py install
```

1.2 Events

sugarcrm_user_creation_succeeded CRM user has been created.

sugarcrm_user_update_succeeded CRM user has been updated.

sugarcrm_user_deletion_succeeded CRM user has been deleted.

sugarcrm_user_activation_succeeded CRM user has been activated.

sugarcrm_user_deactivation_succeeded CRM user has been de-activated.

CHAPTER 2

API

2.1 SugarCRM service settings

SugarCRM service settings have additional quotas:

- sugarcrm_user_count - total count of user limits that were connected to SPLs.

2.2 SugarCRM services list

To get a list of services, run GET against **/api/sugarcrm/** as authenticated user.

2.3 Create a SugarCRM service

To create a new SugarCRM service, issue a POST with service details to **/api/sugacrm/** as a customer owner.

Request parameters:

- name - service name,
- customer - URL of service customer,
- settings - URL of SugarCRM settings, if not defined - new settings will be created from server parameters,
- dummy - is service dummy,

The following rules for generation of the service settings are used:

- **backend_url - URL of template group that describes OpenStack instance provision with default parameters** (required, e.g.: http://example.com/api/template-groups/16c7675752244f5d9e870a2cb0cfeb02/);
- username - NodeConductor user username (e.g. User);
- password - NodeConductor user password (e.g. Password);
- license_code - License code that will be used for SugarCRM activation (required);
- **user_data - User data that will be passed to CRMs OpenStack instance on creation.** Word {password} will be replaced with auto-generated admin password (default: "#cloud-config:nruncmd:n - [bootstrap, -p, {password}]");
- protocol - CRMs access protocol. (default: "http");
- phone_regex - RegEx for phone validation;

- sms_email_from - Name of SMS email sender (SMS will not be send without this parameter);
- sms_email_rcpt - Name of SMS email recipient (SMS will not be send without this parameter);

Example of a request:

```
POST /api/sugarcrm/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "name": "My SugarCRM"
    "customer": "http://example.com/api/customers/2aadad6a4b764661add14dfdda26b373/",
    "backend_url": "http://example.com/api/template-groups/16c7675752244f5d9e870a2cb0cfeb02/",
    "username": "User",
    "password": "Password",
    "license_code": "some-code"
}
```

2.4 Link service to a project

In order to be able to provision SugarCRM resources, it must first be linked to a project. To do that, POST a connection between project and a service to **/api/sugarcrm-service-project-link/** as staff user or customer owner. For example,

```
POST /api/sugarcrm-service-project-link/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "project": "http://example.com/api/projects/e5f973af2eb14d2d8c38d62bcbaccb33/",
    "service": "http://example.com/api/sugarcrm/b0e8a4cbd47c4f9ca01642b7ec033db4/"
}
```

To remove a link, issue DELETE to url of the corresponding connection as staff user or customer owner.

2.5 Project-service connection list

To get a list of connections between a project and an oracle service, run GET against **/api/sugarcrm-service-project-link/** as authenticated user. Note that a user can only see connections of a project where a user has a role.

2.6 Service-project-link quotas

- user_limit_count - limitation for total number of users in all CRMs.
- crm_count - CRMs count.

2.7 Create a new SugarCRM resource

A new SugarCRM instance can be created by users with project administrator role, customer owner role or with staff privilege (is_staff=True). To create a CRM, client must issue POST request to **/api/sugarcrm-crms/** with parameters:

- name - CRM name;
- description - CRM description (optional);
- link to the service-project-link object;
- user_count - maximal number of users in CRM (default: 10);

Example of a valid request:

```
POST /api/sugarcrm-crms/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "name": "test CRM",
    "description": "sample description",
    "service_project_link": "http://example.com/api/sugarcrm-service-project-link/1/",
    "size": 1024,
    "user_count": 20
}
```

2.8 Updating a SugarCRM resource

SugarCRM can be update by issuing PUT request against **/api/sugarcrm-crms/<crm_uuid>/**.

Supported fields for update are **name** and **description**. Quota management for users is performed through **/api/quotas/** endpoint by POSTing a new limit for the **user_count** quota with scope of the SugarCRM instance.

2.9 SugarCRM resource display

To get SugarCRM resource data issue GET request against **/api/sugarcrm-crms/<crm_uuid>/**. Field "instance_url" is visible only for staff.

Example rendering of the CRM object:

```
[
    {
        "url": "http://example.com/api/sugarcrm-crms/7693d9308e0641baa95720d0046e5696/",
        "uuid": "7693d9308e0641baa95720d0046e5696",
        "name": "test-sugarcrm",
        "description": "",
        "start_time": "2015-10-19T08:06:15Z",
        "service": "http://example.com/api/sugarcrm/655b79490b63442d9264d76ab9478f62/",
        "service_name": "sugarcrm service",
        "service_uuid": "655b79490b63442d9264d76ab9478f62",
        "project": "http://example.com/api/projects/0e86f04bb1fd48e181742d0598db69d5/",
        "project_name": "sugarcrm project",
        "project_uuid": "0e86f04bb1fd48e181742d0598db69d5",
```

```
        "customer": "http://example.com/api/customers/3b0fc2c0f0ed4f40b26126dc9cbd8f9f/",
        "customer_name": "sugarcrm customer",
        "customer_native_name": "",
        "customer_abbreviation": "",
        "project_groups": [],
        "resource_type": "SugarCRM.CRM",
        "state": "Provisioning",
        "created": "2015-10-20T10:35:19.146Z",
        "instance_url": "http://example.com/api/openstack-instances/42c35f288c524d52b86d945adde91db2/
        "api_url": "http://example.com",
        "publishing_state": "not published",
        "quotas": [
            {
                "url": "http://example.com/api/quotas/224c771110cc4340aa6a18f58861b307/",
                "uuid": "224c771110cc4340aa6a18f58861b307",
                "name": "user_count",
                "limit": 10.0,
                "usage": 0.0,
            }
        ]
    }
]
```


2.10 Delete CRM

To delete CRM - issue DELETE request against **/api/sugarcrm-crms/<crm_uuid>/**.

2.11 List CRM users

To get list of all registered on CRM users - issue GET request against **/api/sugarcrm-crms/<crm_uuid>/users/**. Only users with view access to CRM can view CRM users.

Supported filters:

- ?user_name
- ?first_name
- ?last_name
- ?status - the status can be Active, Inactive or Reserved.

Response example:


```
[
    {
        "url": "http://example.com/api/sugarcrm-crms/24156c367e3a41eea81e374073fa1060/users/a67a5b55-
        "uuid": "a67a5b55-bb5f-1259-60a2-562e3c88fb34",
        "user_name": "user",
        "status": "Active",
        "last_name": "User",
        "first_name": "",
        "email": "user@example.com"
    }
]
```

2.12 Create new CRM user

To create new CRM user - issue POST request against **/api/sugarcrm-crms/<crm_uuid>/users/**.

Request parameters:

- user_name - new user username;
- last_name - new user last name;
- first_name - new user first name (can be empty);
- email - new user email (can be empty);
- phone - new user mobile phone number (can be empty);
- status - new user status (can be empty);

Example of a request:

```
POST /api/sugarcrm/24156c367e3a41eea81e374073fa1060/users/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "user_name": "test_user",
    "last_name": "test user last name"
}
```

2.13 Update a CRM user

To update CRM user - issue PATCH request against **/api/sugarcrm-crms/<crm_uuid>/users/<user_id>/**.

Example of a request:

```
PUT /api/sugarcrm/24156c367e3a41eea81e374073fa1060/users/cc420109-a419-3d5b-558b-567168cf750f/ HTTP/1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "email": "test_user@example.com",
}
```

2.14 Delete a CRM user

To delete CRM user - issue DELETE request against **/api/sugarcrm-crms/<crm_uuid>/users/<user_id>/**.

2.15 Reset user password

To reset user password - issue POST request against **/api/sugarcrm-crms/<crm_uuid>/users/<user_id>/password/**. You can specify *notify* parameter in order to send user notification about newly created password.

Example of a valid request:

```
POST /api/sugarcrm-crms/db82a52368ba4957ac2cdb6a37d22dee/users/cc420109-a419-3d5b-558b-5671/password/
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "notify": "true"
}
```

Example of response:

{ “password”: “uONLv0UjcI”

}

CHAPTER 3

Endpoints

3.1 NodeConductor SugarCRM

NodeConductor SugarCRM

3.1.1 /api/sugarcrm/

A filter backend that uses django-filter. Supported actions and methods:

/api/sugarcrm/

Methods: `GET`, `POST`
Supported fields for creation:

- **name** – `string`
- project – `link to /api/projects/<uuid>/`
- **customer** – `link to /api/customers/<uuid>/`
- **settings** – `link to /api/service-settings/<uuid>/`
- backend_url – `URL` (URL of template group that describes OpenStack instance provision with default parameters(required, e.g.: http://example.com/api/template-groups/16c7675752244f5d9e870a2cb0cfeb02/))
- username – `string` (NodeConductor user username (e.g. Username))
- password – `string` (NodeConductor user password (e.g. Password))
- available_for_all – `boolean` (Service will be automatically added to all customers projects if it is available for all)
- scope – `link to any: /api/sugarcrm-crms/<uuid>/, /api/openstack-instances/<uuid>/, /api/openstack-tenants/<uuid>/, /api/openstack-volumes/<uuid>/, /api/openstack-snapshots/<uuid>/, /api/openstack-dr-backups/<uuid>/` (VM that contains service)
- protocol – `string` (CRMs access protocol (default: "http"))
- sms_email_rcpt – `string` (Name of SMS email recipient)
- sms_email_from – `string` (Name of SMS email sender)
- license_code – `string` (License code that will be used for SugarCRM activation)
- user_data – `string` (User data that will be passed to CRMs OpenStack instance on creationWord {password} will be replaced with auto-generated admin password (default: "#cloud-config:nruncmd:n - [bootstrap, -p, {password}, -k, {license_code}, -v]"))
- phone_regex – `string` (RegEx for phone validation)

Filter fields:

- ?customer = `UUIDFilter`
- ?name = `string`
- ?settings = `link`
- ?project_uuid = `UUIDFilter`
- ?project = `link`
- ?tag = `ModelMultipleChoiceField`
- ?rtag = `ModelMultipleChoiceField`
- ?shared = `boolean`
- ?type = `ServiceTypeFilter`

To list all services without regard to its type, run **GET** against */api/services/* as an authenticated user.
To list services of specific type issue **GET** to specific endpoint from a list above as a customer owner. Individual endpoint used for every service type.
To create a service, issue a **POST** to specific endpoint from a list above as a customer owner. Individual endpoint used for every service type.
You can create service based on shared service settings. Example:

```
POST /api/digitalocean/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "name": "Common DigitalOcean",
    "customer": "http://example.com/api/customers/1040561ca9e046d2b74268600c7e1105/",
    "settings": "http://example.com/api/service-settings/93ba615d6111466ebe3f792669059cb1/"
}
```

Or provide your own credentials. Example:

```
POST /api/oracle/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com
```

/api/sugarcrm/<uuid>/

Methods: GET, PUT, PATCH, DELETE
Supported fields for update:

- **name** – string
- available_for_all – boolean (Service will be automatically added to all customers projects if it is available for all)

/api/sugarcrm/<uuid>/link/

Methods: GET, POST
To get a list of resources available for import, run **GET** against */<service_endpoint>/link/* as an authenticated user. Optionally project_uuid parameter can be supplied for services requiring it like OpenStack.
To import (link with NodeConductor) resource issue **POST** against the same endpoint with resource id.

```
POST /api/openstack/08039f01c9794efc912f1689f4530cf0/link/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
    "backend_id": "bd5ec24d-9164-440b-a9f2-1b3c807c5df3",
    "project": "http://example.com/api/projects/e5f973af2eb14d2d8c38d62bcbaccb33/"
}
```

/api/sugarcrm/<uuid>/managed_resources/

Methods: GET

/api/sugarcrm/<uuid>/unlink/

Methods: POST
Unlink all related resources, service project link and service itself.

3.1.2 /api/sugarcrm-service-project-link/

A filter backend that uses django-filter. Supported actions and methods:

/api/sugarcrm-service-project-link/

Methods: `GET`, `POST`
Supported fields for creation:

- **project** – `link to /api/projects/<uuid>/`
- **service** – `link to /api/sugarcrm/<uuid>/`

Filter fields:

- ?project = `link`
- ?service_uuid = `UUIDFilter`
- ?customer_uuid = `UUIDFilter`
- ?project_uuid = `UUIDFilter`

To get a list of connections between a project and an service, run **GET** against service_project_link_url as authenticated user. Note that a user can only see connections of a project where a user has a role.
If service has *available_for_all* flag, project-service connections are created automatically. Otherwise, in order to be able to provision resources, service must first be linked to a project. To do that, **POST** a connection between project and a service to service_project_link_url as stuff user or customer owner.

/api/sugarcrm-service-project-link/<pk>/

Methods: `GET`, `DELETE`
To remove a link, issue **DELETE** to URL of the corresponding connection as stuff user or customer owner.

3.1.3 /api/sugarcrm-crms/

SLA filter

Allows to filter or sort resources by actual_sla Default period is current year and month.

Example query parameters for filtering list of OpenStack instances:

```
/api/openstack-instances/?actual_sla=90&period=2016-02
```

Example query parameters for sorting list of OpenStack instances:

```
/api/openstack-instances/?o=actual_sla&period=2016-02
```

Monitoring filter

Filter and order resources by monitoring item. For example, given query dictionary

```
{
    'monitoring__installation_state': True
}
```

it produces following query

```
{
    'monitoring_item__name': 'installation_state',
    'monitoring_item__value': True
}
```

Example query parameters for sorting list of OpenStack instances:

```
/api/openstack-instances/?o=monitoring__installation_state
```

Tags ordering. Filtering for complex tags.

Example: ?tag__license-os=centos7 - will filter objects with tag "license-os:centos7".

Allow to define next parameters in view:

- tags_filter_db_field - name of tags field in database. Default: tags.
- tags_filter_request_field - name of tags in request. Default: tag.

In PostgreSQL NULL values come *last* with ascending sort order. In MySQL NULL values come *first* with ascending sort order. This filter provides unified sorting for both databases. Supported actions and methods:

/api/sugarcrm-crms/

Methods: `GET`, `POST`
Supported fields for creation:

- **name** – `string`
- description – `string`
- **service_project_link** – `link to /api/sugarcrm-service-project-link/<pk>/`
- user_count – `integer`

Filter fields:

- ?customer = `UUIDFilter`
- ?customer_native_name = `string`
- ?project_name = `string`
- ?service_settings_name = `string`
- ?project_group = `UUIDFilter`
- ?project_uuid = `UUIDFilter`
- ?service_uuid = `UUIDFilter`
- ?service_settings_uuid = `UUIDFilter`
- ?customer_uuid = `UUIDFilter`
- ?uuid = `UUIDFilter`
- ?customer_abbreviation = `string`
- ?name = `string`
- ?project = `UUIDFilter`
- ?state = `choice('Deleting', 'Deletion Scheduled', 'Erred', 'Offline', 'Online', 'Provisioning', 'Provisioning Scheduled', 'Resizing', 'Resizing Scheduled', 'Restarting', 'Restarting Scheduled', 'Starting', 'Starting Scheduled', 'Stopping', 'Stopping Scheduled')`
- ?tag = `ModelMultipleChoiceField`
- ?project_group_uuid = `UUIDFilter`
- ?rtag = `ModelMultipleChoiceField`
- ?service_name = `string`
- ?project_group_name = `string`
- ?description = `string`
- ?customer_name = `string`

Order fields: `created`, `customer_abbreviation`, `customer_name`, `customer_native_name`, `name`, `project_group_name`, `project_name`, `state`

/api/sugarcrm-crms/<uuid>/

Methods: GET, PUT, PATCH, DELETE
Supported fields for update:

- **name** – string
- description – string

Optional *field* query parameter (can be list) allows to limit what fields are returned. For example, given request /api/openstack-instances/<uuid>/?field=uuid&field=name you get response like this:

```
{
    "uuid": "90bcfe38b0124c9bbdadd617b5d739f5",
    "name": "Azure Virtual Machine"
}
```

/api/sugarcrm-crms/<uuid>/unlink/

Methods: POST

/api/sugarcrm-crms/<crm_uuid>/users/

Methods: GET, POST, PUT, PATCH, DELETE

/api/sugarcrm-crms/<crm_uuid>/users/<pk>/

Methods: GET, POST, PUT, PATCH, DELETE

/api/sugarcrm-crms/<crm_uuid>/users/<pk>/password/

Methods: POST

CHAPTER 4

License

NodeConductor SugarCRM plugin is open-source under MIT license.

CHAPTER 5

Indices and tables

- genindex
- search

S