
NodeConductor JIRA Documentation

Release 0.4.0

OpenNode

September 16, 2016

1	Guide	3
2	API	5
3	License	13
4	Indices and tables	15

Plugin for interaction and management of Atlassian JIRA.

1.1 Installation

- Install NodeConductor
- Clone NodeConductor JIRA repository

```
git clone https://github.com/opennode/nodeconductor-jira.git
```

- Install NodeConductor JIRA into NodeConductor virtual environment

```
cd /path/to/nodeconductor-jira/  
python setup.py install
```

1.2 Configuration

NodeConductor can integrate with Atlassian JIRA to provide support to the end-users.

Expected structure for the JIRA project is as follows:

- Existing issue type: Support Request (must be default issue type for the project)
- Custom fields:
 - Impact, type: Text Field (single line)
 - Original Reporter, type: Text Field (single line)

Expected permissions:

Permission	Permission code
Add Comments	COMMENT_ISSUE
Edit Own Comments	COMMENT_EDIT_OWN
Browse Projects	BROWSE

1.3 WebHook Setup

It's possible to track updates of JIRA issues and apply them to NodeConductor immediately.

An instruction of JIRA configuration can be found at <https://developer.atlassian.com/jiradev/jira-apis/webhooks>

WebHook URL should be defined as *http://nodeconductor.example.com/api/jira-webhook-receiver/* and following events enabled:

- issue created
- issue updated
- issue deleted

1.4 Example Setup

1.4.1 1. Create support service

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

{
  "name": "JIRA Support",
  "customer": "http://example.com/api/customers/eea999ddf31540aea6bd4f591aa353d1/",
  "backend_url": "https://jira.example.com/",
  "username": "username",
  "password": "password",
  "available_for_all": false
}
```

1.4.2 2. Import support project

Make sure custom fields configured properly and “available_for_all” property is set to true.

```
POST /api/jira/a2f322fed8c444fab48547f595b34279/link/ HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Token c84d653b9ec92c6cbac41c706593e66f567a7fa4
Host: example.com

{
  "backend_id": "SPT",
  "project": "http://example.com/api/projects/e63838e3e68f4fc4aa39617b7550cef3/",
  "impact_field": "Impact",
  "reporter_field": "Original Reporter",
  "default_issue_type": "Support Request",
  "available_for_all": true
}
```

1.4.3 3. Perform support actions

Please use a project created about to post issues.

2.1 JIRA

JIRA

2.1.1 `/api/jira/`

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

/api/jira/

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 (JIRA host (e.g. <https://jira.example.com/>))
- **username** – string (JIRA user with excessive privileges)
- **password** – string
- **available_for_all** – boolean (Service will be automatically added to all customers projects if it is available for all)
- **scope** – link to any: /api/jira-projects/<uuid>/ (VM that contains service)

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/93ba615d6111466ebe3f792669059cb/"
}
```

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

{
  "name": "My Oracle",
  "customer": "http://example.com/api/customers/1040561ca9e046d2b74268600c7e1105/",
  "backend_url": "https://oracle.example.com:7802/em",
  "username": "admin",
  "password": "secret"
}
```

/api/jira/<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/jira/<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/jira/<uuid>/managed_resources/

Methods: GET

/api/jira/<uuid>/unlink/

Methods: POST

Unlink all related resources, service project link and service itself.

2.1.2 /api/jira-webhook-receiver/

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

/api/jira-webhook-receiver/

Methods: POST

Supported fields for creation:

- **webhookEvent** – choice('jira:issue_created', 'jira:issue_deleted', 'jira:issue_updated')
- **timestamp** – UnixTimeField
- **changelog** – JiraChangelogSerializer
- **comment** – JiraCommentSerializer
- **issue** – JiraIssueSerializer
- **user** – JiraUserSerializer

2.1.3 /api/jira-service-project-link/

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

/api/jira-service-project-link/

Methods: GET, POST

Supported fields for creation:

- **project** – link to /api/projects/<uuid>/
- **service** – link to /api/jira/<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/jira-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.**2.1.4 /api/jira-projects/****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/jira-projects/

Methods: GET, POST

Supported fields for creation:

- **name** - string
- **description** - string
- **service_project_link** - link to `/api/jira-service-project-link/<pk>/`
- **key** - string
- **impact_field** - string
- **reporter_field** - string
- **default_issue_type** - string
- **available_for_all** - boolean (Allow access to any user)

Filter fields:

- `?available_for_all = boolean`

/api/jira-projects/<uuid>/

Methods: GET, PUT, PATCH, DELETE

Supported fields for update:

- **name** – string
- **description** – string
- **impact_field** – string
- **reporter_field** – string
- **default_issue_type** – string
- **available_for_all** – boolean (Allow access to any user)

2.1.5 /api/jira-issues/

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

/api/jira-issues/

Methods: GET, POST

Supported fields for creation:

- **user** – link to /api/users/<uuid>/
- **project** – link to /api/jira-projects/<uuid>/
- **summary** – string
- **description** – string
- **priority** – choice('Critical', 'Major', 'Minor', 'n/a')
- **impact** – choice('Large - Whole organization or all services are affected', 'Medium - One department or service is affected', 'Small - Partial loss of service, one person affected', 'n/a')

Filter fields:

- ?status = string
- ?description = string
- ?project_key = string
- ?summary = string
- ?key = string
- ?user_uuid = UUIDFilter

Order fields: created, updated

/api/jira-issues/<uuid>/

Methods: GET, PUT, PATCH, DELETE

Supported fields for update:

- **user** – link to /api/users/<uuid>/
- **summary** – string
- **description** – string
- **priority** – choice('Critical', 'Major', 'Minor', 'n/a')
- **impact** – choice('Large - Whole organization or all services are affected', 'Medium - One department or service is affected', 'Small - Partial loss of service, one person affected', 'n/a')

2.1.6 /api/jira-comments/

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

/api/jira-comments/

Methods: GET, POST

Supported fields for creation:

- **issue** – link to /api/jira-issues/<uuid>/
- **message** – string

Filter fields:

- ?issue_key = string
- ?user_uuid = UUIDFilter
- ?issue_uuid = UUIDFilter

/api/jira-comments/<uuid>/

Methods: GET, PUT, PATCH, DELETE

Supported fields for update:

- **message** – string

2.1.7 /api/jira-attachments/

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

/api/jira-attachments/

Methods: GET, POST

Supported fields for creation:

- **issue** – link to /api/jira-issues/<uuid>/
- **file** – file

Filter fields:

- ?issue_key = string

/api/jira-attachments/<uuid>/

Methods: GET, PUT, PATCH, DELETE

Supported fields for update:

- **file** – file

License

NodeConductor JIRA plugin is open-source under MIT license.

Indices and tables

- `genindex`
- `search`