Siemens AG

National Geothermal Data System  
Software Developer’s Guide

Version: V1.0

Christoph Kuhmuench, Paul Bruschi, Monica McKenna

Siemens Corporate Research

Stephen Richard

Arizona Geological Survey

Edit History

Most recent edit: 2014-04-02 08:40

|  |  |  |  |
| --- | --- | --- | --- |
| Version: | Date: | Author: | Details |
| 0.1 | Roberto Silva Filho | 05/28/2013 | Initial Draft Created |
| 0.2 | Monica McKenna | 06/11/2013 | Minor updates |
| 0.3 | Monica McKenna | 06/24/2013 | Combining comments from a few people |
| 0.4 | Monica McKenna | 06/25/2013 | Added appendix with summary of development.ini changes |
| 0.5 | Monica McKenna | 07/22/2013 | A little re-organization, more hints, and added gdal |
| 0.6 | Monica McKenna | 07/24/2013 | Updating with feedback |
| 0.7 | Christoph Kuhmuench | 12/26/2013 | Updating to latest installer. |
|  | Stephen Richard | 01/28/2014 | review, edit |

Table of Contents

[1 Preface 2](#_Toc377482188)

[1.1 Purpose and Audience 2](#_Toc377482189)

[1.2 Document Roadmap 2](#_Toc377482190)

[1.3 Document Scope and Background 3](#_Toc377482191)

[2 Configure System for Development 4](#_Toc377482192)

[2.1 Stopping Apache2 4](#_Toc377482193)

[2.2 Stopping the Celeryd (Optional) 5](#_Toc377482194)

[2.3 Configure CKAN for development mode 5](#_Toc377482195)

[3 Relevant Frameworks 7](#_Toc377482196)

[4 Understanding CKAN Routes 8](#_Toc377482197)

[5 Remote Debugging the node in a box and central deployments on the amazon machine 18](#_Toc377482198)

[5.1 Prerequisites 18](#_Toc377482199)

[5.2 Outline 18](#_Toc377482200)

[5.3 Local Steps 18](#_Toc377482201)

[5.3.1 PyCharm 18](#_Toc377482202)

[5.3.1.1 Configuring a remote python interpreter 18](#_Toc377482203)

[5.3.1.2 Creating a debug configuration 19](#_Toc377482204)

[5.3.2 SSH Port Forwarding 19](#_Toc377482205)

[5.4 Remote Steps 19](#_Toc377482206)

[5.5 Testing the setup 20](#_Toc377482207)

[6 FAQs and gotchas 21](#_Toc377482208)

[6.1.1 How do I compile the project's less files to css files and minify js resources? 21](#_Toc377482209)

[6.1.2 How do I run a paster command? 21](#_Toc377482210)

[6.1.3 As soon as I complete installing NGDS, when I navigate to the library page, I get a 404 - Group not found. What do I do? 21](#_Toc377482211)

List of Figures

[Figure 1: NGDS is a grid of repositories 3](#_Toc377482212)

# Preface

The National Geothermal Data System (NGDS) supports the storage and search of information resources relevant to the discovery, understanding, and utilization of geothermal energy. It is network of data providers supplying data and metadata, with a aggregating that provides a single entry point for searching resources avaialble through the system.

Nodes in this system focus on one of two roles:

* publishing: nodes that are primarily used by a data provide to create metadata, make files avaialble for network access, deploy NGDS services to provide data access, or make metadata available for harvesting
* aggregating: nodes that are primarily used to collect metadata from NGDS publishing nodes (and possibly from other metadata sources), and provide search and data browsing services to help users find what they need, evaluate it, and get it for their application.

## Purpose and Audience

This document is a step by step tutorial to guide new developers and users to setup both publishing and aggregating NGDS nodes. It is intended for a technical reader who need to understand the concepts and the reasoning of the installation process. Targeted audience includes:

* NGDS System Administrator
* Software Architects
* **Software Developers**
* DoE Monitors

The purpose of the document is to help NGDS system administrators and software developers working with the NGDS code base to become productive quickly. This documentation is also intended to provide background guidance for the organization that has architectural oversight of the software. Othe design and build project product

## Document Roadmap

We assume that the reader is familiar with the NGDS Requirements and Architecture documents. This document outlines the architecture of NGDS and is structured in the following way:

1. Configure the production version for development
2. Relevant frameworks and links to their description
3. Configure Eclipse to be used with CKAN
4. Debugging with Eclipse
5. Highlevel guide line through the code

## Document Scope and Background

The requirements of NGDS are discussed in detail in the requirements document [REQ2.7]. The software architecture is described in detail in the software architecture document [ARC1.0]. Software installation is discussed in detail in the installaion guide [INS1.0].

As outlined in the architecture document the NGDS software stack is the same for a publishing installation and an aggregating installation (see Figure 1). The install process includes a configuration parameter that allows to choose between installation as a Publisher (Node-in-a-Box) or as an aggregater (Central Harvesting Node).

Figure 1: NGDS is a grid of repositories

# Configure System for Development

NOTE:

The NGDS developer GitHub WIKI at <https://github.com/ngds/dev-info/wiki/_pages> includes the most up to date information about sofware development for the NGDS CKAN application.

In this section we outline how to modify the system so that it can be used in development mode. We assume that the installation has been completed as outlined in the Installation Guide [INS1.0].

The following figure outlines the main components of NGDS.

The NGDS CKAN project is based on CKAN 2.0.1. The current method of tracking CKAN is to bind the sofware installaion to a specific versioned GitHub URL of CKAN (as opposed to forking the project). The installation script specifically uses https://github.com/okfn/ckan.git@ckan-2.0.1#egg=ckan

Software development is possible in production configuration with CKAN served indirectly via an apache2 web server. However, it is preferable to run the system directly via the command line for software development. Also it is recommended to turn on the development mode in order to get more feedback and because CKAN serves source JavaScript files instead of compressed JavaScript files in this mode. The installer script configures the [celeryd](http://www.celeryproject.org/) demon to run as a service, which is generally ok from the developers perspective, but we also describe how to shutdown the service if that is desired.

## Stopping Apache2

In order to run the system in debugging mode first shut down apache2:

% sudo service apache2 stop

This command shuts the apache server down; the server will restart when you restart the system. To deactivate the apache service so it does not automatically restart, execute the following command:

% sudo sh -c "echo 'manual' > /etc/init/apache2.override"

## Stopping the Celeryd service (Optional)

For most developers it is convenient to run the celeryd distributed task queue as a service. If there is a need to deactivate and run the service manually this is how it is done:

% sudo service ngds-celeryd stop

As before apache2 the service will restart when the system is rebooted. This can be prevented with:

% sudo sh -c "echo 'manual' > /etc/init/ngds-celeryd.override"

## Configure CKAN for development mode

In order to run CKAN in development mode, start the application from the command line via a [paster](http://ckan.readthedocs.org/en/ckan-1.4.1/paster.html) command, after executing step 2.1 above to stop apache2. You can leave the celery demon running.

All NGDS source code can be found under (if the standard installation instructions were followed):

/opt/ngds/bin/default

To run the system from the command line do the following:

% cd /opt/ngds/bin/default

% . ./bin/activate

% cd ckan

% cp /opt/ngds/etc/ckan/default/production.ini ./development.ini

% paster serve development.ini

After this command is executed, NGDS CKAN starts, and the service can be reached at:

<http://127.0.0.1:5000/>

Next, modify the following entry in the development.ini file:

* [line017] Debug= true

This is flag causes two things:

1. CKAN generates debugging output
2. CKAN serves uncondensed JavaScript resources

The uncondensed JavaScript output is especially useful for developers. If the flag is not set, Java­Script resources must be compiled after every change using the command:

% paster --plugin=ckanext-ngds ngdsapi compile\_client\_scripts

Nothing else in development.ini needs to be changed.

# Relevant Frameworks

Please refer to the architecture document to get an overview of all frameworks.

# Understanding CKAN Routes

This is a list of URLs that CKAN listens to. For each URL, I've printed out the "Controller" and "Action" that is performed when a request comes in to that URL. This provides you with a starting point for understanding what CKAN does at any particular page or API call. "Controller" provides a pointer to a class, and "Action" is the name of a function inside that class.

If the Controller is not fully qualified, then they exist at ckan.controllers.{whatever}, for example "home" really means "ckan.controllers.home"

/error/%(action)s

Controller: error

Action: None

/error/%(action)s/%(id)s

Controller: error

Action: None

%(url)s

Controller: home

Action: cors\_options

/api/2/search/%(register)s/geo

Controller: ckanext.spatial.controllers.api:ApiController

Action: spatial\_query

/package/%(id)s/map

Controller: None

Action: None

/dataset/%(id)s/map

Controller: ckanext.spatial.controllers.view:ViewController

Action: wms\_preview

/proxy

Controller: ckanext.spatial.controllers.view:ViewController

Action: proxy

/hello/world/%(eggs)s

Controller: ckanext.helloworld.controller:HelloWorldController

Action: dispatch\_get

/

Controller: home

Action: index

/about

Controller: home

Action: about

/api%(ver)s/action/%(logic\_function)s

Controller: api

Action: action

/api%(ver)s

Controller: api

Action: get\_api

/api%(ver)s/search/%(register)s

Controller: api

Action: search

/api%(ver)s/tag\_counts

Controller: api

Action: tag\_counts

/api%(ver)s/rest

Controller: api

Action: index

/api%(ver)s/qos/throughput/

Controller: api

Action: throughput

/api%(ver)s/rest/%(register)s

Controller: api

Action: list

/api%(ver)s/rest/%(register)s

Controller: api

Action: create

/api%(ver)s/rest/%(register)s/%(id)s

Controller: api

Action: show

/api%(ver)s/rest/%(register)s/%(id)s

Controller: api

Action: update

/api%(ver)s/rest/%(register)s/%(id)s

Controller: api

Action: update

/api%(ver)s/rest/%(register)s/%(id)s

Controller: api

Action: delete

/api%(ver)s/rest/%(register)s/%(id)s/%(subregister)s

Controller: api

Action: list

/api%(ver)s/rest/%(register)s/%(id)s/%(subregister)s

Controller: api

Action: create

/api%(ver)s/rest/%(register)s/%(id)s/%(subregister)s/%(id2)s

Controller: api

Action: create

/api%(ver)s/rest/%(register)s/%(id)s/%(subregister)s/%(id2)s

Controller: api

Action: show

/api%(ver)s/rest/%(register)s/%(id)s/%(subregister)s/%(id2)s

Controller: api

Action: update

/api%(ver)s/rest/%(register)s/%(id)s/%(subregister)s/%(id2)s

Controller: api

Action: delete

/api%(ver)s/util/user/autocomplete

Controller: api

Action: user\_autocomplete

/api%(ver)s/util/is\_slug\_valid

Controller: api

Action: is\_slug\_valid

/api%(ver)s/util/dataset/autocomplete

Controller: api

Action: dataset\_autocomplete

/api%(ver)s/util/tag/autocomplete

Controller: api

Action: tag\_autocomplete

/api%(ver)s/util/resource/format\_autocomplete

Controller: api

Action: format\_autocomplete

/api%(ver)s/util/resource/format\_icon

Controller: api

Action: format\_icon

/api%(ver)s/util/group/autocomplete

Controller: api

Action: group\_autocomplete

/api%(ver)s/util/markdown

Controller: api

Action: markdown

/api%(ver)s/util/dataset/munge\_name

Controller: api

Action: munge\_package\_name

/api%(ver)s/util/dataset/munge\_title\_to\_name

Controller: api

Action: munge\_title\_to\_package\_name

/api%(ver)s/util/tag/munge

Controller: api

Action: munge\_tag

/api%(ver)s/util/status

Controller: api

Action: status

/api%(ver)s/util/snippet/%(snippet\_path)s

Controller: api

Action: snippet

/api%(ver)s/i18n/%(lang)s

Controller: api

Action: i18n\_js\_translations

/packages

Controller: None

Action: None

/packages/%(url)s

Controller: None

Action: None

/package

Controller: None

Action: None

/package/%(url)s

Controller: None

Action: None

/dataset/%(id)s/related/new

Controller: related

Action: new

/dataset/%(id)s/related/edit/%(related\_id)s

Controller: related

Action: edit

/dataset/%(id)s/related/delete/%(related\_id)s

Controller: related

Action: delete

/dataset/%(id)s/related

Controller: related

Action: list

/apps/%(id)s

Controller: related

Action: read

/apps

Controller: related

Action: dashboard

/dataset

Controller: package

Action: search

/dataset/%(action)s

Controller: package

Action: None

/dataset/%(action)s/%(id)s/%(revision)s

Controller: package

Action: read\_ajax

/dataset/%(action)s/%(id)s

Controller: package

Action: None

/dataset/activity/%(id)s/%(offset)s

Controller: package

Action: activity

/dataset/%(id)s.%(format)s

Controller: package

Action: read

/dataset/%(id)s

Controller: package

Action: read

/dataset/%(id)s/resource/%(resource\_id)s

Controller: package

Action: resource\_read

/dataset/%(id)s/resource\_delete/%(resource\_id)s

Controller: package

Action: resource\_delete

/dataset/%(id)s/resource\_edit/%(resource\_id)s

Controller: package

Action: resource\_edit

/dataset/%(id)s/resource/%(resource\_id)s/download

Controller: package

Action: resource\_download

/dataset/%(id)s/resource/%(resource\_id)s/embed

Controller: package

Action: resource\_embedded\_dataviewer

/dataset/%(id)s/resource/%(resource\_id)s/viewer

Controller: package

Action: resource\_embedded\_dataviewer

/dataset/%(id)s/resource/%(resource\_id)s/preview/%(preview\_type)s

Controller: package

Action: resource\_datapreview

/groups

Controller: None

Action: None

/groups/%(url)s

Controller: None

Action: None

/group

Controller: group

Action: index

/group/list

Controller: group

Action: list

/group/new

Controller: group

Action: new

/group/%(action)s/%(id)s

Controller: group

Action: None

/group/activity/%(id)s/%(offset)s

Controller: group

Action: activity

/group/%(id)s

Controller: group

Action: read

/tags

Controller: None

Action: None

/tags/%(url)s

Controller: None

Action: None

/tag/read/%(url)s

Controller: None

Action: None

/tag

Controller: tag

Action: index

/tag/%(id)s

Controller: tag

Action: read

/users/%(url)s

Controller: None

Action: None

/user/

Controller: None

Action: None

/user/edit

Controller: user

Action: edit

/user/activity/%(id)s/%(offset)s

Controller: user

Action: activity

/user/activity/%(id)s

Controller: user

Action: activity

/dashboard/%(offset)s

Controller: user

Action: dashboard

/dashboard

Controller: user

Action: dashboard

/user/follow/%(id)s

Controller: user

Action: follow

/user/unfollow/%(id)s

Controller: user

Action: unfollow

/user/followers/%(id)s

Controller: user

Action: followers

/user/edit/%(id)s

Controller: user

Action: edit

/user/reset/%(id)s

Controller: user

Action: perform\_reset

/user/register

Controller: user

Action: register

/user/login

Controller: user

Action: login

/user/\_logout

Controller: user

Action: logout

/user/logged\_in

Controller: user

Action: logged\_in

/user/logged\_out

Controller: user

Action: logged\_out

/user/logged\_out\_redirect

Controller: user

Action: logged\_out\_page

/user/reset

Controller: user

Action: request\_reset

/user/me

Controller: user

Action: me

/user/set\_lang/%(lang)s

Controller: user

Action: set\_lang

/user/%(id)s

Controller: user

Action: read

/user

Controller: user

Action: index

/revision

Controller: revision

Action: index

/revision/edit/%(id)s

Controller: revision

Action: edit

/revision/diff/%(id)s

Controller: revision

Action: diff

/revision/list

Controller: revision

Action: list

/revision/%(id)s

Controller: revision

Action: read

/feeds/group/%(id)s.atom

Controller: feed

Action: group

/feeds/tag/%(id)s.atom

Controller: feed

Action: tag

/feeds/dataset.atom

Controller: feed

Action: general

/feeds/custom.atom

Controller: feed

Action: custom

/ckan-admin

Controller: admin

Action: index

/ckan-admin/%(action)s

Controller: admin

Action: None

/api/storage

Controller: ckan.controllers.storage:StorageAPIController

Action: index

/api/storage/metadata/%(label)s

Controller: ckan.controllers.storage:StorageAPIController

Action: set\_metadata

/api/storage/metadata/%(label)s

Controller: ckan.controllers.storage:StorageAPIController

Action: get\_metadata

/api/storage/auth/request/%(label)s

Controller: ckan.controllers.storage:StorageAPIController

Action: auth\_request

/api/storage/auth/form/%(label)s

Controller: ckan.controllers.storage:StorageAPIController

Action: auth\_form

/storage/upload

Controller: ckan.controllers.storage:StorageController

Action: upload

/storage/upload\_handle

Controller: ckan.controllers.storage:StorageController

Action: upload\_handle

/storage/upload/success

Controller: ckan.controllers.storage:StorageController

Action: success

/storage/upload/success\_empty

Controller: ckan.controllers.storage:StorageController

Action: success\_empty

/storage/f/%(label)s

Controller: ckan.controllers.storage:StorageController

Action: file

/i18n/strings\_%(lang)s.js

Controller: util

Action: i18n\_js\_strings

/util/redirect

Controller: util

Action: redirect

/testing/primer

Controller: util

Action: primer

/testing/markup

Controller: util

Action: markup

/stats

Controller: ckanext.stats.controller:StatsController

Action: index

/stats/%(action)s

Controller: ckanext.stats.controller:StatsController

Action: None

# Remote Debugging NGDS CKAN nodes deployed on the Amazon cloud

## Prerequisites

* PyCharm Professional Edition
* The NGDS project setup on PyCharm. (Importing the pyenv directory into pycharm is the ideal setup)
* Ability to ssh to the Amazon machine
* Either Linux, or Putty on Windows. (users have had problems with OpenSSH on Windows)

## Outline

1. Create configuration entries for a remote python interpreter (Amazon cloud machine), a remote debug configuration that uses the remote python interpreter to serve our application.
2. Create a mapping between local and remote source directories so breakpoints set on the local source code will be reflected on the remote instance.
3. Create an ssh tunnel and forward a local port to the Amazon machine so that we can navigate the UI locally on our browser by setting a socks proxy variable in our network configuration.
4. Create a copy of the configuration file, one for each instance we're debugging, and modify it to change the port number that the application will be served on.

We'd like to leave everything else the same so that we have all the data and indexes that the actual instance has.

## Local Steps

### PyCharm

#### Configuring a remote python interpreter

* Go to Settings, and find the Python Interpreters section.
* Click on the + button and select remote in the dropdown.

The values that go into the dialog should look like this -

Host : 1.1.1.1 Port : 22

Username : ubuntu

Auth Type : Keypair (OpenSSH)

Private Key File : /path/to/my/priv\_key.txt

Passphrase : abracadabraismypassphrase

// The passphrase you typed in when you generated your private key.

Python Interpreter Path : /home/ubuntu/pyenv/bin/python

// This is the path to the virtual environment's python binary. This should correspond to the instance you're debugging.

* Click on Test Connection to verify that your settings are accurate.
* You'll do this once for every instance you'd like to debug, for example, one for the central instance and one for the node instance.

#### Creating a debug configuration

* Click on the Run menu item and click on 'Edit Configurations'
* Click on the + button to add a new debug configuration entry.

The values that go in there should somewhat like this

Name : Remote Node-in-a-box

Script : bin/paster serve src/ckan/debug.ini --reload

// Assuming you set your working directory to /home/ubuntu/pyenv

Python Interpreter : Remote Python

// Select the interpreter you created in the earlier step

Working directory : /home/ubuntu/pyenv/

Path Mappings : /home/ubuntu/pyenv - /home/mylocalusername/mylocalpyenv

* Click Apply.

### SSH Port Forwarding

* Open up a terminal, type in ssh -D 7777 ubuntu@1.1.1.1 -v
* Substitute 1.1.1.1 with the IP of the remote machine.
* Open up your browser, I prefer firefox since you can apply the proxy to only firefox and not modify the system's network configuration.
* Navigate to Advanced > Network Settings and select the SOCKS Proxy item and enter in localhost and port 7777 into the Host and Port entries respectively.

## Remote Steps

* SSH to the remote machine
* Navigate to the ckan directory of the instance you want to debug.
* Create a copy of development.ini in the same directory and call it debug.ini. cp development.ini debug.ini
* Modify development.ini and in the [server:main] section, modify the port number and set it to 7777 (or some other value that is not being used. You have to make sure this is the same as the port you entered in the SSH port forwarding step)

## Testing the setup

* In pycharm, select the debug configuration we just created and click on the debug button (or select debug from the tools menu).
* The console readout will indicate an ssh connection being made, and if it's successful, will tell you which port the application is being served on.
* Now open up a terminal and type in ssh -D 7777 ubuntu@1.1.1.1 -v (Replace 1.1.1.1 with the IP of the machine you're debugging)
* Navigate to localhost:7777 on firefox. If all the previous steps were done correctly, you should see the NGDS UI come up.

# FAQs and gotchas

### How do I compile the project's *less* files to *css* files and minify js resources?

A) Run the paster command

paster --plugin=ckanext-ngds ngdsapi compile\_client\_scripts from the ckan installation directory.

### How do I run a paster command?

A) Paster commands are declared in setup.py files. For this reason, they are always to be run from the directory containing the setup.py file that declares that command.

CKAN commands are run in two ways - either

cd /path/to/src/ckan  
paster --plugin=ckanext-ngds ngdsapi doc-index #to run an ngds command.

Or,

cd /path/to/ckanext/ngds  
paster ngdsapi doc-index --c /path/to/src/ckan/development.ini

The first command is run from the ckan installation directory where it's development.ini and setup.py file are to be found. The second command is run from the ckanext-ngds/ckanext/ngds directory where it's setup.py file declaring the ngdsapi command is to be found, and the path to ckan's development.ini is provided. The only reason for this is that ckan (and ckan plugin) commands are run against a live ckan instance.

### As soon as I complete installing NGDS, when I navigate to the library page, I get a 404 - Group not found. What do I do?

A) You'll need to create an organization called public. Open up development/production.ini and remove ngdsui from the plugin list (it is to be found under ckan.plugins). Restart CKAN and then navigate to the organization page and create an organization with the title set to 'public'. Then add ngdsui back into the plugins list and restart ckan. You should now be able to see the library page with no errors.