
gs.config Documentation

Release 2.3.0

GroupServer.org

April 28, 2016

1	gs.config API Reference	3
1.1	Configuration	3
1.2	Errors	5
2	Example	7
2.1	File	7
2.2	Parsing	7
3	Changelog	9
3.1	2.3.0 (2015-03-30)	9
3.2	2.2.0 (2015-03-17)	9
3.3	2.1.3 (2014-09-16)	9
3.4	2.1.2 (2014-06-24)	9
3.5	2.1.1 (2014-05-05)	9
3.6	2.1.0 (2014-04-25)	9
3.7	2.0.0 (2013-08-10)	10
3.8	1.0.0 (2012-07-13)	10
4	Resources	11
5	Indices and tables	13
	Python Module Index	15

This package provides some classes that are used to manage “sets” of configuration options.

Contents:

gs.config API Reference

The package exports the following API symbols.

1.1 Configuration

class `gs.config.Config`(*configset*, *configpath*=None)
The configuration.

Method `Config`(*configset*, *configpath*=None)

Parameters

- **configset** (*str*) – The name of the configuration set to read.
- **configpath** (*str*) – The path to the configuration file. If None *and* Zope is being used then some standard Zope configuration directories are checked (see `Config.get_zope_config()`).

Raises

- `gs.config.errors.ConfigPathError` – No path could be found.
- `gs.config.errors.ConfigFileError` – The configuration file could not be read.
- `gs.config.errors.ConfigSetError` – The configuration file lacks *configset*.

The actual parsing of the configuration file is done by the `ConfigParser` module.

get (*configtype*, *strict*=True)
Get the values defined in a section

Parameters

- **configtype** (*str*) – The ID for the section to retrieve.
- **strict** (*bool*) – If True (the default) then a `ConfigNoOption` error is raised when an option is present in the configuration file but absent from the schema. When False the option is ignored.

Returns The values for all options in the provided section.

Return type A dict containing *optionId*: *value* pairs. The values are coerced using the schema set by `set_schema()`.

Raises

- `gs.config.errors.ConfigNoSectionError` – No section for the ID in configtype exists.
- `gs.config.errors.ConfigNoOptionError` – An option was present in the configuration file but absent from the section.
- `gs.config.errors.ConfigConvertError` – An option could not be coerced.

Example:

```
smtpConfig = config.get('smtp')
```

get_schema (configtype)

Get the schema that is currently set for a section.

Parameters `configtype (str)` – The identifier for the section.

Returns The schema that is currently set for the section.

Return type A dict, containing optionId: type pairs.

Raises `gs.config.errors.ConfigNoSchemaError` – No schema with the identifier configtype found.

static get_zope_config ()

Try and figure out where the groupserver config is, using Zope

Returns The location of the config file from Zope.

Return type str

Raises `gs.config.errors.ConfigFileError` – The configuration file failed to be read.

The location of the configuration file should either be the `$ZOE_HOME/etc/gsconfig.ini` or `$INSTANCE_HOME/etc/gsconfig.ini`, with the latter preferred for backwards compatibility, but the former being more typical. This normally equates to `etc/gsconfig.ini` within what the installation documentation refers to as *the GroupServer directory*.

keys ()

Get the list of sections that are currently defined.

Returns A list of sections for the configuration set.

Return type list

set_schema (configtype, schema)

Set the schema that is used for parsing the options.

Parameters

- **configtype (str)** – The identifier for the section of the configuration.
- **schema (dict)** – The schema for the section, as optionId: type pars.

When the value for an option in a section is retrieved its *type* is coerced from a string to one of the types passed in as schema.

Example:

```
s = {'station': str,
     'frequency': int,}
conf.set_schema('radio', s)
```

gs.config.getInstanceId ()

Get the ID of the current instance.

Returns The ID the of the instance, or `default`

Return type `str`

It can be useful to have multiple *instances* running on one server, but configured separately. The `getInstanceId` function gets the ID of the current instance by looking it up in the `HTTP_INSTANCEID` property of the current Zope HTTP request. If Zope (and HTTP) are not being used then `default` is returned.

This function is defined in `gs.config` mostly out of convinience, as `GroupServer` normally organises its configuration into *instances*.

1.2 Errors

The possible configuration errors.

exception `gs.config.errors.ConfigConvertError`

An error raised when the value cannot be converted.

exception `gs.config.errors.ConfigError`

A generic error with the configuration.

exception `gs.config.errors.ConfigFileError`

An error with reading the config file

exception `gs.config.errors.ConfigNoOptionError`

An error raised when an option is not defined in a schema

exception `gs.config.errors.ConfigNoSchemaError`

An error raised when there is no schema

exception `gs.config.errors.ConfigNoSectionError`

An error raised when there is no configuration section specified.

exception `gs.config.errors.ConfigPathError`

An error with the path to the config file.

exception `gs.config.errors.ConfigSetError`

An error with the structure of the config file.

Example

Below is an example of a *file* with configuration sets, and *parsing* the file.

2.1 File

In the following example the default configuration-set contains two items, smtp and pop. The the configuration for this set is provided by the smtp-local and pop-local sections. The smtp-remote section is currently unused by the configuration set.

```
[config-default]
smtp = local
pop = local

[smtp-local]
server = localhost
port = 2525

[pop-local]
server = localhost
port = 110

[smtp-remote]
server = smtp.example.com
port = 25
```

2.2 Parsing

A configuration class is initialised. The second parameter is optional, depending on the degree to which we want the environment to configure things automatically.

```
>>> from gs.config import Config
>>> config = Config('default', '/example/file.ini')
```

A schema must be provided before data is retrieved. For example, setting the server to be a string, and a port to be an integer.

```
>>> config.set_schema('smtp', {'server': str, 'port': int})
```

Then a specific configuration section, with all the options can be retrieved as a dict.

```
>>> config.get('smtp')
{'port': 2525, 'server': localhost}
```

If file fails to fit the schema then a ConfigError is raised:

```
>>> config.set_schema('smtp', {'someparam': int})
>>> config.get('smtp')
```

```
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "config.py", line 201, in get
    raise ConfigNoOptionError(msg)
ConfigNoOptionError: No option "server" defined in schema for "smtp".
```

However, it is possible to parse the configuration in a lax way, by passing `strict=False`. This allows for hard-coded defaults:

```
>>> config.get('smtp', strict=False)
{}
```

Changelog

3.1 2.3.0 (2015-03-30)

- Checking both the instance home and the Zope-home for the configuration file

3.2 2.2.0 (2015-03-17)

- Allowing the parsing of the configuration file to be lax

3.3 2.1.3 (2014-09-16)

- Renaming the `*txt` files as `*rst` files, for [GitHub](#)

3.4 2.1.2 (2014-06-24)

- Fixing a coding error, where a `m` was used in an error, rather than a `msg`

3.5 2.1.1 (2014-05-05)

- Added Sphinx documentation

3.6 2.1.0 (2014-04-25)

- Adding Python 3 support
- Updating documentation
- Adding unit tests and tox support

3.7 2.0.0 (2013-08-10)

- Merge of some code from `gs.database`
- Reduced dependency on Zope (now an extra requirement)

3.8 1.0.0 (2012-07-13)

- Initial version

Resources

- Documentation: <http://groupserver.readthedocs.io/projects/gsconfig>
- Code repository: <https://github.com/groupserver/gs.config>
- Questions and comments to <http://groupserver.org/groups/development>
- Report bugs at <https://redmine.iopen.net/projects/groupserver>

Indices and tables

- `genindex`
- `modindex`
- `search`

g

`gs.config.errors`, 5

C

Config (class in `gs.config`), 3
ConfigConvertError, 5
ConfigError, 5
ConfigFileError, 5
ConfigNoOptionError, 5
ConfigNoSchemaError, 5
ConfigNoSectionError, 5
ConfigPathError, 5
ConfigSetError, 5

G

`get()` (`gs.config.Config` method), 3
`get_schema()` (`gs.config.Config` method), 4
`get_zope_config()` (`gs.config.Config` static method), 4
`getInstanceId()` (in module `gs.config`), 4
`gs.config.errors` (module), 5

K

`keys()` (`gs.config.Config` method), 4

S

`set_schema()` (`gs.config.Config` method), 4