

---

# Bugzilla Documentation

*Release 0.2*

**David Burns**

July 20, 2014



<b>1</b>	<b>Installing Bugsy</b>	<b>3</b>
<b>2</b>	<b>Using Bugsy</b>	<b>5</b>
2.1	Getting a bug from Bugzilla . . . . .	5
2.2	Creating a new bug . . . . .	5
2.3	Searching Bugzilla . . . . .	5
<b>3</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Python Module Index</b>	<b>13</b>



Bugsy is a tool that allows you to programmatically work with Bugzilla using its native REST API.

To use you will do

```
import bugsy
bugzilla = bugsy.Bugsy()
bug = bugzilla.get(123456)
bug123456.status = 'RESOLVED'
bug123456.resolution = 'FIXED'
bugzilla.put(bug123456)
```



---

## Installing Buggy

---

To install Buggy, simply use pip or easy install

Pip

```
pip install buggy
```

easy\_install

```
easy_install buggy
```





---

## Using Buggy

---

### 2.1 Getting a bug from Bugzilla

Getting a bug is quite simple. Create a Buggy object and then get the bug number that you want.

```
import buggy
bugzilla = buggy.Bugzilla()
bug = bugzilla.get(123456)
```

### 2.2 Creating a new bug

To create a new bug, create a Bug object, populate it with the items that you need and then use the Buggy object to put the bug into Bugzilla

```
import buggy
bug = buggy.Bug()
bug.summary = "I really really love cheese"
bug.add_comment("and I really want sausages with it!")

bugzilla = buggy.Bugzilla("username", "password")
bugzilla.put(bug)
bug.id #returns the bug id from Bugzilla
```

### 2.3 Searching Bugzilla

To search for bugs you will need to create a Buggy object and then you can call `search_for` and chain the search. The Search API is a [Fluent API](#) so you just chain the items that you need and then call `search` when the search is complete.

```
import buggy
bugzilla = buggy.Bugzilla()
bugs = bugzilla.search_for\
    .keywords("checkin-needed")\
    .include_fields("flags")\
    .search()
```

More details can be found in from the Search class

To see further details look at:

### 2.3.1 Buggy

**class** `bugsy.Bugsy` (*username=None, password=None, bugzilla\_url='https://bugzilla.mozilla.org/rest'*)

Bugsy allows easy getting and putting of Bugzilla bugs

**get** (*bug\_number*)

Get a bug from Bugzilla. If there is a login token created during object initialisation it will be part of the query string passed to Bugzilla

**Parameters** **bug\_number** – Bug Number that will be searched. If found will return a Bug object.

```
>>> bugzilla = Buggy()
>>> bug = bugzilla.get(123456)
```

**put** (*bug*)

This method allows you to create or update a bug on Bugzilla. You will have had to pass in a valid username and password to the object initialisation and received back a token.

**Parameters** **bug** – A Bug object either created by hand or by using `get()`

If there is no valid token then a `BugsyException` will be raised. If the object passed in is not a Bug then a `BugsyException` will be raised.

```
>>> bugzilla = Buggy()
>>> bug = bugzilla.get(123456)
>>> bug.summary = "I like cheese and sausages"
>>> bugzilla.put(bug)
```

**request** (*path, method='GET', \*\*kwargs*)

Perform a HTTP request.

Given a relative Bugzilla URL path, an optional request method, and arguments suitable for `requests.Request()`, perform a HTTP request.

**class** `bugsy.BugsyException` (*msg*)

If while interacting with Bugzilla and we try do something that is not supported this error will be raised.

**class** `bugsy.LoginException` (*msg*)

If a username and password are passed in but we don't receive a token then this error will be raised.

### 2.3.2 Bug

**class** `bugsy.Bug` (*bugsy=None, \*\*kwargs*)

This represents a Bugzilla Bug

**OS**

Property for getting or setting the OS that the bug occurred on

```
>>> bug.OS
"All"
```

**add\_comment** (*comment*)

Adds a comment to a bug. If a bug does not have a bug ID then you need call `put` on the `Bugsy` class.

```
>>> bug.add_comment("I like sausages")
>>> bugzilla.put(bug)
```

If it does have a bug id then this will do a post to the server

```
>>> bug.add_comment("I like eggs too")
```

**component**

Property for getting the bug component

```
>>> bug.component
General
```

**get\_comments()**

Obtain comments for this bug.

Returns a list of Comment instances.

**id**

Property for getting the ID of a bug.

```
>>> bug.id
123456
```

**platform**

Property for getting the bug platform

```
>>> bug.platform
"ARM"
```

**product**

Property for getting the bug product

```
>>> bug.product
Core
```

**resolution**

Property for getting or setting the bug resolution

```
>>> bug.resolution
"FIXED"
```

**status**

Property for getting or setting the bug status

```
>>> bug.status
"REOPENED"
```

**summary**

Property for getting and setting the bug summary

```
>>> bug.summary
"I like cheese"
```

**to\_dict()**

Return the raw dict that is used inside this object

**update()**

Update this object with the latest changes from Bugzilla

```
>>> bug.status
'NEW'
#Changes happen on Bugzilla
>>> bug.update()
>>> bug.status
'FIXED'
```

**version**

Property for getting the bug platform

```
>>> bug.version
"TRUNK"
```

**class** `bugsy.BugException(msg)`

If we try do something that is not allowed to a bug then this error is raised

### 2.3.3 Search

Changed in version 0.2.

**class** `bugsy.Search(bugsy)`

This allows searching for bugs in Bugzilla

**assigned\_to** (\*args)

When search() is called it will search for bugs assigned to these users

**Parameters** `args` – items passed in will be turned into a list

**Returns** `Search`

```
>>> bugzilla.search_for.assigned_to("dburns@mozilla.com")
```

**bug\_number** (`bug_numbers`)

When you want to search for a bugs and be able to change the fields returned.

**Parameters** `bug_numbers` – A string for the bug number or a list of strings

**Returns** `Search`

```
>>> bugzilla.search_for.bug_number(['123123', '123456'])
```

**include\_fields** (\*args)

Include fields is the fields that you want to be returned when searching. These are in addition to the fields that are always included below.

**Parameters** `args` – items passed in will be turned into a list

**Returns** `Search`

```
>>> bugzilla.search_for.include_fields("flags")
```

**The following fields are always included in search:** 'version', 'id', 'summary', 'status', 'op\_sys', 'resolution', 'product', 'component', 'platform'

**keywords** (\*args)

When search() is called it will search for the keywords passed in here

**Parameters** `args` – items passed in will be turned into a list

**Returns** `Search`

```
>>> bugzilla.search_for.keywords("checkin-needed")
```

**search** ()

Call the Bugzilla endpoint that will do the search. It will take the information used in other methods on the Search object and build up the query string. If no bugs are found then an empty list is returned.

```
>>> bugs = bugzilla.search_for\  
...         .keywords("checkin-needed")\  
...         .include_fields("flags")\  
...         .search()
```

**summary** (\*args)

When search is called it will search for bugs with the words passed into the methods

**Parameters** **args** – items passed in will be turned into a list

**Returns** `Search`

```
>>> bugzilla.search_for.summary("663399")
```

**whiteboard** (\*args)

When search is called it will search for bugs with the words passed into the methods

**Parameters** **args** – items passed in will be turned into a list

**Returns** `Search`

```
>>> bugzilla.search_for.whiteboard("affects")
```



---

## Indices and tables

---

- *genindex*
- *modindex*
- *search*





## **b**

bugsy, [8](#)