BugConnect

Corey Butler

Chief Consultant
Ecor Systems, LLC
http://www.ecorsystems.com

May 02, 2009

Table of Contents

About BugConnect

This connector is a modified subset of a larger API originally authored by Corey Butler of Ecor Systems. It provides basic connectivity and several methods to simplify interaction between ColdFusion and Bugzilla. The code is being released under the Apache Version 2 license.

The package consists of 3 ColdFusion components and a configuration file. The code follows a loose object-oriented approach and does not rely on any framework. It was tested with **`Bugzilla 3.2.3`** and **`3.3.4`**[1]. Each Bugzilla instance used in testing was running on a **`PostgreSQL 8.3`** database, but it should work with MySQL as well.

Intended Use, Purpose, & Development Approach

This package was **not** designed to be a replacement for the Bugzilla administration screens. It is meant to assist developers with the creation of simpler/cleaner public interfaces. It was designed with the expectation developers will still handle bug maintenance as they normally do[2], but end users will not know Bugzilla is being used on the backend.

Why create this?

We originally developed a beta testing website hosted on an internet-accessible server and wanted to use our existing Bugzilla system to track issues submitted by our testers. However; the Bugzilla server was hosted on our private local network & we didn't like the presentation capabilities of Bugzilla. This package was quickly developed to bridge the gap.

What approach was used to develop this?

We didn't want to use the Bugzilla XML-RPC web services but feared it would take too long to write the code necessary to generate a bug and all of its relationships via SQL, especially since Bugzilla 3.3.4 has an updated data model & is subject to change in future editions. Therefore we took the approach of posting forms to the Bugzilla server. It is the slowest option we faced, but we really didn't notice the Bugzilla servers taking too long to respond.

Basic SQL queries are used to retrieve data, but were generally avoided for any other purpose. There are, however, a few circumstances where data is inserted directly to the database. For example, Bugzilla requires a security token in the posted form to match a value in the database. We inserted this token with `CFQUERY` and used it in the form posted to the Bugzilla server. Direct SQL is also used to create/remove database user accounts. The only major area where a SQL `INSERT` statement is used is when adding a comment to a bug (**`bug.cfc → addComment()`**) or adding a user to the CC list (**`bug.cfc → addCCUser()`**).

What's NOT included?

There wasn't time to include search methods. Hopefully this will be coming in a future edition.

[1] Bugzilla 3.3.4 is in beta as of this writing.
[2] We use Eclipse+Mylyn to connect to streamline development with Bugzilla.

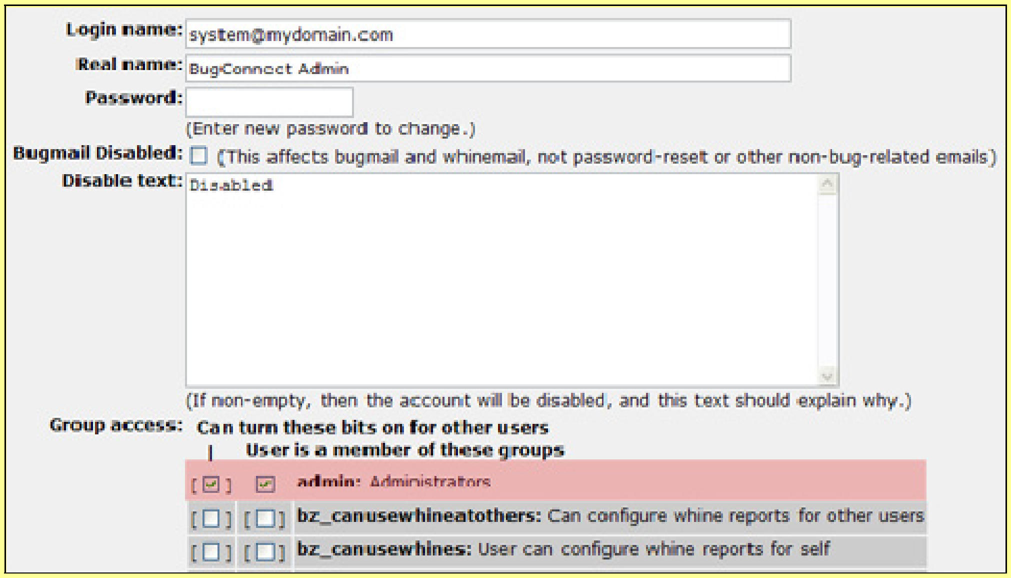
Installation & Examples

First, install the package by copying the `com` directory to your application root or create a mapping in the CF administrator. The bug and product CFCs both extend the factory, so you'll need to change the `<cfcomponent extends="com.utility.bugzilla.factory"…` in each of those files if you use a different location.

If you haven't already, create a DSN in the ColdFusion admin pointing to your Bugzilla database.

Copy the configuration file to a location ColdFusion can read. It may be a good idea to put this on a non web-accessible drive or use a ColdFusion mapping. Bugzilla requires an administrative user to create or remove other users, and unfortunately this must be stored in plain text.

It is *strongly recommended* that a dedicated *database* Bugzilla user be created specifically for use with this connector. This is especially true if your Bugzilla instance leverages LDAP or Active Directory because this will avoid storing any LDAP/AD password in the configuration. You can disable the Bugzilla user in the Bugzilla administration screens by providing text in the **`Disabled Text`** section.



Login name: system@mydomain.com
Real name: BugConnect Admin
Password:
(Enter new password to change.)
Bugmail Disabled: ☐ (This affects bugmail and whinemail, not password-reset or other non-bug-related emails)
Disable text: Disabled
(If non-empty, then the account will be disabled, and this text should explain why.)
Group access: Can turn these bits on for other users
| User is a member of these groups
[☑] ☑ admin: Administrators
[☐] [☐] bz_canusewhineatothers: Can configure whine reports for other users
[☐] [☐] bz_canusewhines: User can configure whine reports for self

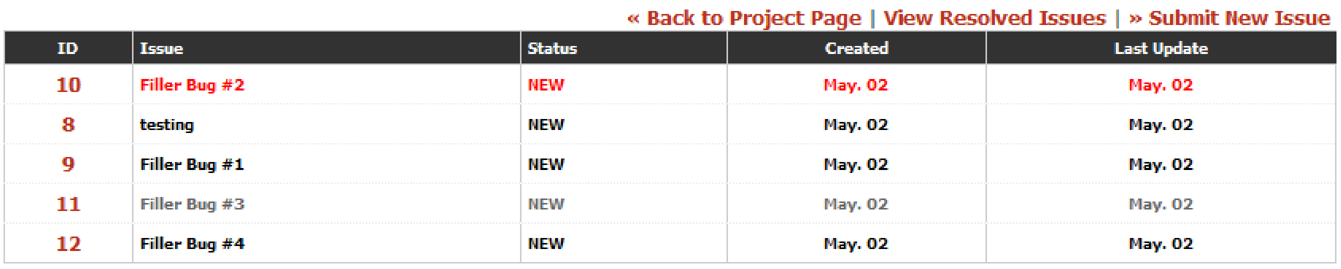
The configuration file is pretty straightforward:

`[bugzilla]`	The "section" argument for **init()**
`server=http://bugs.mydomain.com`	The Bugzilla server
`port=80`	The port Bugzilla is running on
`dsn=bugs`	The CF DSN used to connect to the Bugzilla database.
`pfx=bugzilla.`	The prefix of the Bugzilla database tables.
`user=system@mydomain.com`	The **admin** user (see box above to create)
`password=MyPassword`	The **admin password**.
`priority=P3`	The default priority assigned to a bug.
`severity=normal`	The default severity assigned to a bug.
`status=NEW`	The default status assigned to a bug.
`OS=Other`	The default operating system assigned to a bug when none is known.
`platform=Other`	The default hardware platform assigned to a bug.
`assignment=`	The default user to whom a bug is assigned. This must be a valid Bugzilla user. Leave this blank to use the default user configured within Bugzilla.

Once this is setup, it's safe to begin coding. The following are a few examples of where to begin.

Really Basic Bug Listing

The following screenshot depicts a list of all open bugs for a specific product (#1).



« Back to Project Page | View Resolved Issues | » Submit New Issue

ID	Issue	Status	Created	Last Update
10	Filler Bug #2	NEW	May. 02	May. 02
8	testing	NEW	May. 02	May. 02
9	Filler Bug #1	NEW	May. 02	May. 02
11	Filler Bug #3	NEW	May. 02	May. 02
12	Filler Bug #4	NEW	May. 02	May. 02

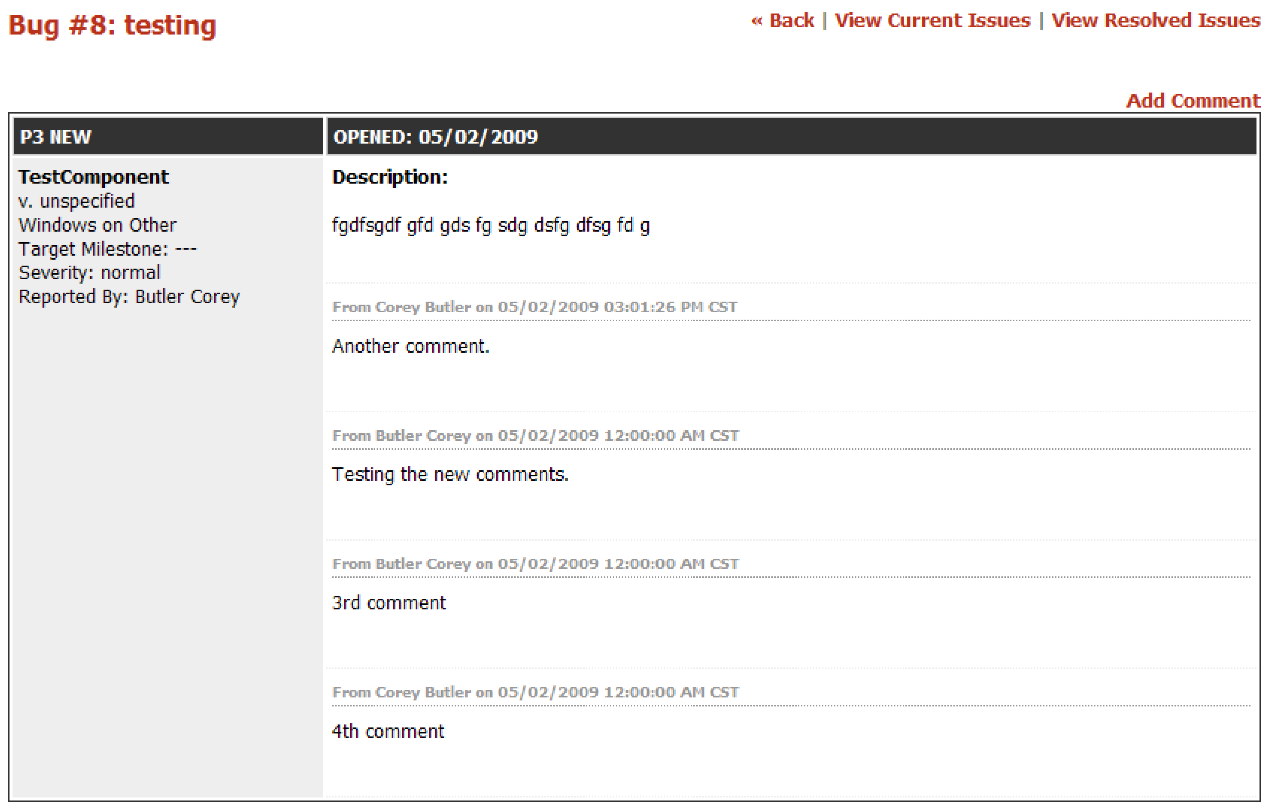
The code used to retrieve the bug listing was:

```
<cfscript>
   prod = createObject("component","com.utility.bugzilla.product");
   prod.init("/path/to/config.ini","bugzilla",1);
   qry = prod.getBugs("UNCONFIRMED,NEW,ASSIGNED,REOPENED,VERIFIED");
<cfscript>

<table cellpadding="4" cellspacing="0" border="0" id="bug">
   <tr>
      <th class="center">ID</th>
      <th>Issue</th>
      <th>Status</th>
      <th class="center">Created</th>
      <th class="center" class="last">Last Update</th>
   </tr>
   <cfoutput query="qry">
   <tr>
      <td class="center">
          <a href="#CGI.PATH_INFO#?bug=#bug_id#">#bug_id#</a>
      </td>
      <td>#short_desc#</td>
      <td>#ucase(bug_status)#</td>
      <td align="center">#DateFormat(creation_ts,"Mmm. dd")#</td>
      <td align="center" class="last">#DateFormat(delta_ts,"Mmm. dd")#</td>
   </tr>
   </cfoutput>
</table>
```

As you can see, this is basically the same as querying the bugs table. In most cases, we simply use it to return a list of bug IDs linking to a page. Then we use the bug object to provide detail and functionality.

Viewing & Managing a Bug

The following screenshot was taken from a newly created test bug.



Bug #8: testing

« Back | View Current Issues | View Resolved Issues

Add Comment

P3 NEW	OPENED: 05/02/2009
TestComponent v. unspecified Windows on Other Target Milestone: --- Severity: normal Reported By: Butler Corey	**Description:** fgdfsgdf gfd gds fg sdg dsfg dfsg fd g From Corey Butler on 05/02/2009 03:01:26 PM CST Another comment. From Butler Corey on 05/02/2009 12:00:00 AM CST Testing the new comments. From Butler Corey on 05/02/2009 12:00:00 AM CST 3rd comment From Corey Butler on 05/02/2009 12:00:00 AM CST 4th comment

```
<cfscript>
   bug = createObject("component","com.utility.bugzilla.bug");
   bug.init("/path/to/config.ini","bugzilla",url.bug);
<cfscript>

<cfoutput>
<table class="bug" cellpadding="4" cellpadding="0" border="0">
   <tr>
      <th width="25%">#bug.priority# #bug.status#</th>
      <th>OPENED: #bug.createdate#</th>
   </tr>
   <tr>
      <td>
         <b>#bug.component#</b><br/>
         v. #bug.version#<br/>
         #bug.os# on #bug.platform#<br/>
         Target Milestone: #bug.targetmilestone#<br/>
         Severity: #bug.severity#<br/>
         Reported By: #bug.reporter#<br/>
         <cfif len(trim(bug.deadline))>
            <br/>
            <i>Deadline:</i> #DateFormat(bug.deadline,"Mmm dd, yyyy")#<br/>
         </cfif>
      </td>
         .
         .
         .
<cfoutput>
```

Comments and CC Lists

Adding comments, adding CC list members, and removing CC list members to a bug is accomplished with a single line of code (each):

```
<cfscript>
   bug = createObject("component","com.utility.bugzilla.bug");
   bug.init("/path/to/config.ini","bugzilla",url.bug);

   //Add Comment, Add CC User, Remove CC User
   bug.addComment('j@doe.com',trim(form.comment));
   bug.addCCUser('rr@acme.com');
   bug.removeCCUser('rr@acme.com');
<cfscript>
```

Creating & Disabling Bugzilla Users

Sometimes you need to create new users. For example, we need to create new users in Bugzilla on the fly when someone new posts an issue for the first time in our public beta site. In our case, they have already registered an account with us, so we create their Bugzilla account automatically.

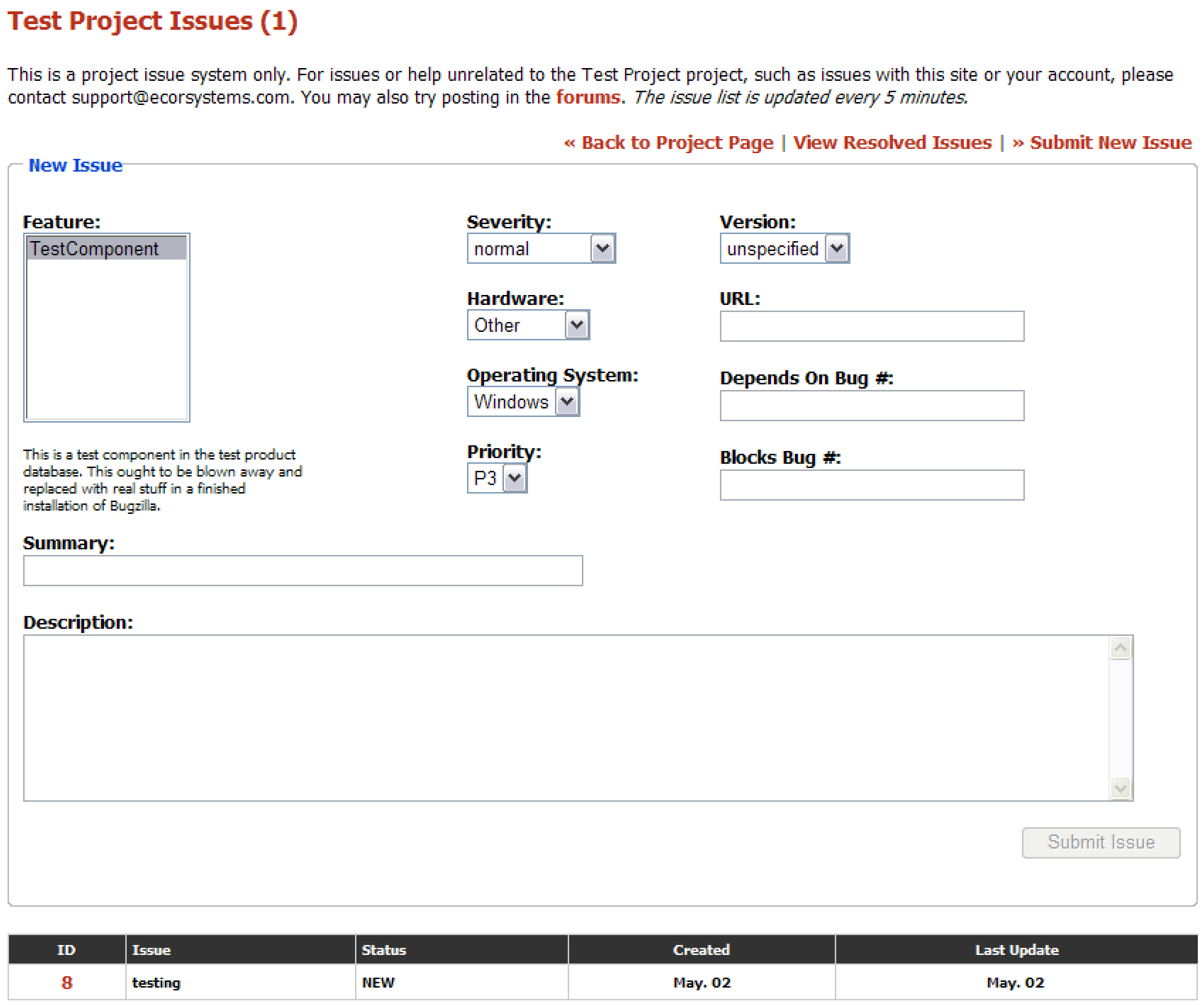
```
<cfscript>
   factory = createObject("component","com.utility.bugzilla.factory");
   factory.init("/path/to/config.ini","bugzilla");

   //Add user to Bugzilla & join them to a product group
   if (not factory.userExists('j@doe.com')) {
      factory.createUser('j@doe.com','passwd','John Doe');
      factory.addUserToGroup('j@doe.com','testgroup');
   }

   //Remove User
   factory.disableUser('j@doe.com');

</cfscript>
```

A note about disabling users:

It's possible to remove users instead of just disabling them, but it is not a recommended practice and therefore not included in the package at this time. It is better to just disable a user in order to maintain the integrity of the Bugzilla database. If a user is removed entirely, there may be complications with the bugs that user is associated with.

Possibilities

This package has provided us with a lot of opportunities to simply use Bugzilla as a service within our ColdFusion applications. The bug entry form shown below is an example of what we wanted to achieve on a regular basis. Using the BugConnect package, this form was created in about 5 minutes and we were seeing the results in Eclipse+Mylyn momentarily after entry.



Test Project Issues (1)

This is a project issue system only. For issues or help unrelated to the Test Project project, such as issues with this site or your account, please contact support@ecorsystems.com. You may also try posting in the **forums**. *The issue list is updated every 5 minutes.*

« Back to Project Page | View Resolved Issues | » Submit New Issue

New Issue

Feature: TestComponent

This is a test component in the test product database. This ought to be blown away and replaced with real stuff in a finished installation of Bugzilla.

Severity: normal

Hardware: Other

Operating System: Windows

Priority: P3

Version: unspecified

URL:

Depends On Bug #:

Blocks Bug #:

Summary:

Description:

Submit Issue

ID	Issue	Status	Created	Last Update
8	testing	NEW	May. 02	May. 02

Conclusion

We hope others find this mini-package useful. This was just a snippet of a larger proprietary package that cannot be released publicly. There is no intention (at this time) to provide active support, so community contributions & support are encouraged. If you'd like to leave feedback or get in touch, please send a message to info@ecorsystems.com or visit http://www.ecorsystems.com.

Documentation

Factory

com.utility.bugzilla.factory

hierarchy:	WEB-INF.cftags.component com.utility.bugzilla.factory
path:	Provides base methods and properties of the Bugzilla installation.
properties:	adminpassword, adminuser, default, dsn, key, options, pfx, port, product, server, url
methods:	addUserToGroup, bugExists, createBug, createUser, disableUser, getBugzillaToken, getProducts*, init, removeAllUserBugzillaTokens, removeBugzillaToken, userExists

* - private method

Property	Hint	Type	Req.	Implemented In	Default Value
adminpassword	The obfuscated password of the account used to login.	string		factory	-
adminuser	The administrative account with permission to add users and query all products.	string		factory	-
default	A struct containing 6 default values (keys): priority, platform, OS, status, assignedTo, and estimatedTime.	string		factory	-
dsn	The DSN used to connect to the Bugzilla DB.	string		factory	-
key	A key used for obfuscating the password.	string		factory	-
options	A struct containing arrays with system options.	string		factory	-
pfx	The prefix of the Bugzilla DB schema and/or table name.	string		factory	-
port	The port on which Bugzilla is running.	string		factory	80
product	A struct containing the product ID's associated with this Bugzilla instance. Each ID is a key containing a sub-struct. The sub-struct keys include name and closed. Closed is a true/false value representing whether or not the product is open for new bug submission.	string		factory	-
server	The core URL of the Bugzilla installation.	string		factory	-
url	A struct containing the URL values	string		factory	-

	of each Bugzilla form handler.				

addUserToGroup

public void **addUserToGroup** (*required string* user, *required numeric* group)

Adds the user to the specified group.

Output: suppressed
Parameters:
user: string, required, user - The email address of the user.
group: numeric, required, group - The ID of the group to add the user to.

bugExists

public boolean **bugExists** (*required numeric* id)

Given a Bug ID, determines whether it exists or not.

Output: suppressed
Parameters:
id: numeric, required, id - The ID of the bug.

createBug

public void **createBug** (*required string* user, *required string* pwd, *required any* exception, *required string* name, *required string* product, *required string* component, *string* version="1.0", *string* platform, *string* os, *string* priority, *string* severity, *string* status, *string* assignedTo, *string* cc="", *string* estimatedTime, *string* location="", *string* deadline="", *string* keywords="", *string* dependson="", *boolean* forceNew="false")

Creates a new bug. If a bug with the same name, location, and version is submitted, the existing bug's vote count will be increased by one.

Output: suppressed
Parameters:
user: string, required, user - The email address of the user. If using LDAP or Active Directory, make sure this is the same email account associated with the user's LDAP/AD account.
pwd: string, required, pwd - A plain text password (will be auto-encrypted).
exception: any, required, exception - The issue to log. This can be any type of data, which will be dumped to the database. For robust error checking, provide a WDDX value.
name: string, required, name - The descriptive name of the issue (summary).
product: string, required, product - The product is which the issue arose.
component: string, required, component - The component of the product in which the issue arose.
version: string, optional, version - The version of the product in which the issue arose.
platform: string, optional, platform - The platform on which the issue occurred.
os: string, optional, os - The Operating System on which the issue occurred.
priority: string, optional, priority - The priority level of the issue.
severity: string, optional, severity - The severity level of the issue.
status: string, optional, status - The status of the issue.
assignedTo: string, optional, assignedTo - Who the bug should be assigned to. Leave blank for default.
cc: string, optional, cc - Who should be cc'd on the issue.
estimatedTime: string, optional, estimatedTime - The estimated time required to fix or review the issue.
location: string, optional, location - The URL where the issue occurred.
deadline: string, optional, deadline - The deadline for completing the fix/review.
keywords: string, optional, keywords - A comma delimited list of keywords.
dependson: string, optional, dependson - The ID of another issue on which this new issue is dependent.
forceNew: boolean, optional, forceNew - Forces the creation of a new bug, even if the name, location,

and version match an existing bug.

createUser

public void **createUser** (*required string* email, *required string* pwd, *required string* nm)

Creates a new database user in Bugzilla. This does not support LDAP/Active Directory user creation.

Output: enabled
Parameters:
email: string, required, email - The email address of the user. If using LDAP or Active Directory, make sure this is the same email account associated with the user's LDAP/AD account.
pwd: string, required, pwd - A plain text password (will be auto-encrypted).
nm: string, required, nm - The user's real name.

disableUser

public **disableUser** (*required string* user)

Disables a user.

Output: suppressed
Parameters:
user: string, required, user - The email address of the user.

getBugzillaToken

package string **getBugzillaToken** (*required string* event, *string* user, *string* type="session", *boolean* noforce="false")

Returns a Bugzilla security token for the specified event/user. If no token exists, one is generated unless the noforce attribute is true.

Output: enabled
Parameters:
event: string, required, event - The Bugzilla event for which a token is necessary.
user: string, optional, user - The user requiring a token. If none is specified, the admin user will be used.
type: string, optional, type - The type of session to be created.
noforce: boolean, optional, noforce - Setting this to true will NOT create a new token, even if one cannot be found. If no token is found and none are generated, the resulting value will be 'NONE'.

getProducts*

private struct **getProducts** ( )

Populates the product attribute.

Output: suppressed

init

public void **init** (*required string* ini, *string* section="default")

Initialize the factory as though it's being used by a specific user.

Output: suppressed
Parameters:
ini: string, required, ini - The absolute path of the ini file.

section: string, optional, section - The section of the ini file to use for this initialization.

removeAllUserBugzillaTokens

public void **removeAllUserBugzillaTokens** (*required string* user)

Removes all security tokens for a particular user.

Output: suppressed
Parameters:
user: string, required, user - The user to clear.

removeBugzillaToken

public void **removeBugzillaToken** (*required string* token)

Removes a security token.

Output: suppressed
Parameters:
token: string, required, token - The token to be removed.

userExists

public boolean **userExists** (*required string* email)

Indicates whether the specified user exists or not.

Output: suppressed
Parameters:
email: string, required, email - The email address of the user.

Product

com.utility.bugzilla.product

hierarchy:	WEB-INF.cftags.component com.utility.bugzilla.factory com.utility.bugzilla.product
path:	C:\WEB\WWW\common\api\coldfusion\com\utility\bugzilla\product.cfcRepresents a Bugzilla product.
properties:	adminpassword, adminuser, classification, classificationID, closed, component, default, defaultmilestone, description, dsn, id, key, maxvotesperuser, milestoneurl, name, options, pfx, port, product, server, url, version, versions, votesperbug, votestoconfirm
methods:	addBug, getBugs, init, reinit, save
inherited methods:	addUserToGroup, bugExists, createBug, createUser, disableUser, getBugzillaToken, removeAllUserBugzillaTokens, removeBugzillaToken, userExists

* - private method

Property	Hint	Type	Req.	Implemented In	Default Value
adminpassword	The obfuscated password of the account used to login.	string		factory	-
adminuser	The administrative account with permission to add users and query all products.	string		factory	-
classification	The bug classification.	string		product	-
classificationID	The numeric ID of the bug classification.	numeric		product	-
closed	Idnicates that the product is closed for issue tracking.	boolean		product	false
component	An struct representing a component of the product.	struct		product	-
default	A struct containing 6 default values (keys): priority, platform, OS, status, assignedTo, and estimatedTime.	string		factory	-
defaultmilestone	The default mileston when none is specified	string		product	---
description	A description of the product.	string		product	-
dsn	The DSN used to connect to the Bugzilla DB.	string		factory	-
id	The DB ID of the product	numeric		product	-
key	A key used for obfuscating the password.	string		factory	-

maxvotesperuser	The maximum number of votes a single user can cast for this product.	numeric		product	-
milestoneurl	A URL pointing to a specific product milestone.	string		product	-
name	The name of the product	string		product	-
options	A struct containing arrays with system options.	string		factory	-
pfx	The prefix of the Bugzilla DB schema and/or table name.	string		factory	-
port	The port on which Bugzilla is running.	string		factory	80
product	A struct containing the product ID's associated with this Bugzilla instance. Each ID is a key containing a sub-struct. The sub-struct keys include name and closed. Closed is a true/false value representing whether or not the product is open for new bug submission.	string		factory	-
server	The core URL of the Bugzilla installation.	string		factory	-
url	A struct containing the URL values of each Bugzilla form handler.	string		factory	-
version	The current version of the product.	string		product	-
versions	A structure object containing all of the product versions. Each key of the struct is the ID of a version. Each key contains a version.	struct		product	-
votesperbug	The maximum number of votes per user per bug.	numeric		product	-
votestoconfirm	The minimum number of votes required to auto-confirm a bug in the product.	numeric		product	-

addBug

```
public void addBug ( required string user, required string pwd, required
String summary, required String description, required string component,
string version="1.0", string platform, string os, string priority, string
severity="", string location="", string dependson="" )
```

Adds a new bug to this product.

Output: suppressed
Parameters:
user: string, required, user - The email address of the user. If using LDAP or Active Directory, make sure this is the same email account associated with the user's LDAP/AD account.
pwd: string, required, pwd - A plain text password (will be auto-encrypted).
summary: String, required, summary - The descriptive summary/title of the bug.
description: String, required, description - The message text explaining the issue.
component: string, required, component - The component of the product in which the issue arose.
version: string, optional, version - The version of the product in which the issue arose.
platform: string, optional, platform - The platform on which the issue occurred.
os: string, optional, os - The Operating System on which the issue occurred.
priority: string, optional, priority - The priority level of the issue.
severity: string, optional, severity - The severity of the issue.
location: string, optional, location - The URL where the issue occurred.
dependson: string, optional, dependson - The ID of another issue on which this new issue is dependent.

getBugs

```
public query getBugs ( string status )
```

Returns the bugs associated with this product as a query.

Output: suppressed
Parameters:
status: string, optional, status - Restrict the results to a specific type of status. Accepts a comma delimited list. Examples include NEW,ASSIGNED,etc.

init

```
public void init ( required string ini, string section="default", numeric id
)
```

Initialize the Bugzilla product object.

Output: suppressed
Parameters:
ini: string, required, ini - The absolute path of the ini file for the Bugzilla installation.
section: string, optional, section - The section of the ini file to use for this initialization.
id: numeric, optional, id - The product ID. This is a numeric value stored in the DB.

reinit

```
public void reinit ( numeric id )
```

Populates the object properties.

Output: suppressed
Parameters:
id: numeric, optional, id - The product ID. This is a numeric value stored in the DB.

save

```
public void save ( )
```

Save the core properties of the product.

Output: suppressed

Bug

com.utility.bugzilla.bug

hierarchy:	WEB-INF.cftags.component com.utility.bugzilla.factory com.utility.bugzilla.bug
path:	C:\WEB\WWW\common\api\coldfusion\com\utility\bugzilla\bug.cfcRepresents a specific bug.
properties:	adminpassword, adminuser, alias, assignedTo, assignedToID, cc, cclistAccessible, changeDate, component, componentID, createDate, deadline, default, description, dsn, estimatedTime, everconfirmed, id, key, keywords, lastdiffed, location, options, OS, pfx, platform, port, priority, product, productID, qa, remainingTime, reporter, reporterAccessible, reporterEmail, reporterID, resolution, server, severity, status, summary, targetMilestone, url, version, votes, whiteboard
methods:	addCCUser, addComment, init, reinit, removeCCUser, updateComments*
inherited methods:	addUserToGroup, bugExists, createBug, createUser, disableUser, getBugzillaToken, removeAllUserBugzillaTokens, removeBugzillaToken, userExists

* - private method

Property	Hint	Type	Req.	Implemented In	Default Value
adminpassword	The obfuscated password of the account used to login.	string		factory	-
adminuser	The administrative account with permission to add users and query all products.	string		factory	-
alias	An alias used to identify the bug.	string		bug	-
assignedTo	Who the bug is assigned to.	string		bug	-
assignedToID	The ID of who the bug is assigned to.	string		bug	-
cc	An array of email addresses notified when the bug changes.	array		bug	-
cclistAccessible	Whether the CC list is accessible or not.	boolean		bug	-
changeDate	The last date the bug was modified.	date		bug	-
component	The component name with the bug.	string		bug	-
componentID	The component ID with the bug.	numeric		bug	-
createDate	The date when the bug is created.	date		bug	-
deadline	The deadline for resolution.	date		bug	-

default	A struct containing 6 default values (keys): priority, platform, OS, status, assignedTo, and estimatedTime.	string		factory	-
description	All of the descriptions/comments associated with the bug. Each element of the array is a struct with the following keys: ID, author, authorID, authorEmail, created, comment, private.	array		bug	-
dsn	The DSN used to connect to the Bugzilla DB.	string		factory	-
estimatedTime	The estimated time to resolution.	numeric		bug	-
everconfirmed	Identifies whether the bug was ever confirmed.	boolean		bug	-
id		numeric		bug	-
key	A key used for obfuscating the password.	string		factory	-
keywords	A list of keywords associated with the bug.	string		bug	-
lastdiffed	The last time the bug was checked for differences	date		bug	-
location	A URL where the bug was found.	string		bug	-
options	A struct containing arrays with system options.	string		factory	-
OS	Operating System	string		bug	-
pfx	The prefix of the Bugzilla DB schema and/or table name.	string		factory	-
platform	The platform upon which the issue was found.	string		bug	-
port	The port on which Bugzilla is running.	string		factory	80
priority	Priority of the fix.	string		bug	-
product	A struct containing the product ID's associated with this Bugzilla instance. Each ID is a key containing a sub-struct. The sub-struct keys include name and closed. Closed is a true/false value representing whether or not the product is open for new bug submission.	string		factory	-
productID	The product which the bug is associated with.	numeric		bug	-

qa	The Quality Assurance contact.	string		bug	-
remainingTime	The time remaining to resolution.	numeric		bug	-
reporter	Who reported the issue.	string		bug	-
reporterAccessible	Whether the reporter is accessible or not.	boolean		bug	-
reporterEmail	The reporter email address.	string		bug	-
reporterID	The reporter ID	numeric		bug	-
resolution	The fix/answer.	string		bug	-
server	The core URL of the Bugzilla installation.	string		factory	-
severity	The severity rating of the bug.	string		bug	-
status	The current status of the bug.	string		bug	-
summary	The descriptive summary of the bug.	string		bug	-
targetMilestone	The target milestone for fix/issue.	string		bug	-
url	A struct containing the URL values of each Bugzilla form handler.	string		factory	-
version	The version of the product/component.	string		bug	-
votes	The total number of votes for this issue.	numeric		bug	-
whiteboard	Whiteboard status.	string		bug	-

addCCUser

public void **addCCUser** (*required string* user)

Add a user to the CC list for this issue.

Output: suppressed
Parameters:
user: string, required, user - The email address of the user.

addComment

public void **addComment** (*required string* author, *required string* comment)

Add a comment to the bug.

Output: suppressed
Parameters:
author: string, required, author - The email address of the author.

comment: string, required, comment - The comment text.

init

public void **init** (*required string* ini, *string* section="default", *numeric* id)

Initialize the bug object.

Output: suppressed
Parameters:
ini: string, required, ini - The absolute path of the ini file for the Bugzilla installation.
section: string, optional, section - The section of the ini file to use for this initialization.
id: numeric, optional, id - The bug ID. This is a numeric value stored in the DB.

reinit

public void **reinit** ( )

Populates the primary attributes of the object.

Output: enabled

removeCCUser

public void **removeCCUser** (*required string* user)

Remove a user from the CC list for this issue.

Output: suppressed
Parameters:
user: string, required, user - The email address of the user.

updateComments*

private void **updateComments** ( )

Updates the description attribute of the object.

Output: suppressed