|  |  |
| --- | --- |
| v0.2.15 | March 2015 |
| **Enversion Administration Guide** |  |
|  | Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at:  <http://www.apache.org/licenses/LICENSE-2.0>  Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  See the License for the specific language governing permissions and limitations under the License.  © Copyright 2015 Snakebite, Inc. |
|  |  | |

Issue Record

The amendments made between approved issues of this document are captured below:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Date | Amendments | Author |
| v0.2.15 | 16-Feb-2015 | Initial revision. | Trent Nelson |

Table 1 Issue Record

Table of Contents

Issue Record 2

List of Figures 5

List of Tables 5

Executive Summary 6

1 What’s New 7

1.1 Blocked File Extensions 7

1.2 Custom Hooks 7

2 Command Reference 9

2.1 Administration Commands 10

2.1.1 evnadmin create 10

2.1.2 evnadmin analyze 10

2.1.3 evnadmin enable 10

2.1.4 evnadmin disable 10

2.1.5 evnadmin status 10

2.2 Customization Commands 10

2.2.1 evnadmin set-repo-component-depth 11

2.2.2 evnadmin get-repo-component-depth 11

2.2.3 evnadmin set-repo-custom-hook-class 11

2.2.4 evnadmin get-repo-custom-hook-class 11

2.3 Configuration Commands 11

2.3.1 evnadmin dump-config 11

2.3.2 evnadmin dump-default-config 11

2.3.3 evnadmin dump-modified-repo-config 11

2.3.4 evnadmin dump-repo-config 11

2.3.5 evnadmin show-possible-config-file-load-order 11

2.3.6 evnadmin show-possible-repo-config-file-load-order 11

2.3.7 evnadmin show-actual-config-file-load-order 11

2.3.8 evnadmin show-actual-repo-config-file-load-order 11

2.3.9 evnadmin show-writable-repo-override-config-filename 11

2.4 Root Management Commands 11

2.4.1 evnadmin show-roots 11

3 Repository Layout: Single vs Multiple Component 12

4 Configuration Reference 13

4.1 [main] 13

4.1.1 blocked-file-extensions-iregex 13

4.1.2 custom-hook-classname 14

5 Errors, Warnings and Notes 15

5.1 Errors 15

5.1.1 BlockedFileExtension 15

List of Figures

Figure 1 Example Custom Hook 7

Figure 2 Sample Output: evnadmin 9

Figure 3 Example Single-Component Layout 12

Figure 4 Example Multiple-Component Layout 12

Figure 5 Configuration: custom-hook-classname 14

List of Tables

Table 1 Issue Record 2

Table 2 What’s New: Blocked File Extensions 7

Table 3 Configuration: blocked-file-extension-iregex 13

Table 4 Error: BlockedFileExtension 15

Executive Summary

Enversion is a server-side hook framework for Subversion that blocks over 80 different types of unwanted or problematic commits. It is ideal in enterprise environments where tight controls must be kept on source code control.

The framework is written in Python, and is compatible with Python 2.6 and 2.7. It runs on Linux and OS X.

It is licensed under the same terms as the Apache Subversion project (Apache 2.0).

The project’s source code and issue tracker is hosted on github.com:

<http://github.com/enversion/enversion>

Releases are made available via git tags, signed by “Trent Nelson <trent@trent.me>" with GPG public key DDCBBC36, which is available from the MIT public key server:

<https://pgp.mit.edu/pks/lookup?op=get&search=0x25D93491DDCBBC36>

# What’s New

This section contains new features and enhancements in reverse chronological order (i.e. newest features are listed first).

## Blocked File Extensions

This change adds support for blocking a commit if it contains a path matching a given regular expression.

|  |  |
| --- | --- |
| Since | 0.2.15 |
| Config | blocked-file-extensions-iregex |
| Issue | <https://github.com/enversion/enversion/issues/28> |
| Initial  Commit | <https://github.com/enversion/enversion/commit/d36734a3af05cf77a07ccfaff1d87c7a436ca189> |
| Tests | <https://github.com/enversion/enversion/blob/master/lib/evn/test/test_blocked_file_extensions.py> |

Table 2 What’s New: Blocked File Extensions

## Custom Hooks

Support for custom hooks was introduced in v0.2.14. A custom hook in this context is a Python class that derives from [evn.custom\_hook.CustomHook](https://github.com/enversion/enversion/blob/master/lib/evn/custom_hook.py#L4) and implements either pre\_commit or post\_commit (or both), e.g.:

class OurCustomHook(evn.custom\_hook.CustomHook):

def pre\_commit(self, commit, \*args, \*\*kwds):

...

def post\_commit(self, commit, \*args, \*\*kwds):

...

Figure 1 Example Custom Hook

|  |  |
| --- | --- |
| Since | 0.2.14 |
| Config |  |
| Issue |  |
| Initial  Commit |  |
| Tests |  |

# Command Reference

Enversion is interacted with via the CLI program evnadmin. This program has been modelled after the Subversion CLI program svnadmin, and thus, shares a similar interface and command line parameters where possible.

When run with no arguments, it will print out the available subcommands, e.g.:

% evnadmin

Type 'evnadmin help <subcommand>' for help on a specific subcommand.

Available subcommands:

analyze

create

debug

disable

disable-remote-debug (drd)

doctest

dump-config (dc)

dump-default-config (ddc)

dump-hook-code (dhc)

dump-modified-repo-config (dmrc)

dump-repo-config (drc)

enable

enable-remote-debug (erd)

find-merges (fm)

fix-hooks (fh)

get-repo-component-depth (grcd)

get-repo-custom-hook-class (grchc)

list-unit-test-classnames (lutc)

purge-evn-props (pep)

root-info (ri)

run-hook (rh)

selftest

set-repo-component-depth (srcd)

set-repo-custom-hook-class (srchc)

show-actual-config-file-load-order (sacflo)

show-actual-repo-config-file-load-order (sarcflo)

show-debug-sessions (sds)

show-possible-config-file-load-order (spcflo)

show-possible-repo-config-file-load-order (sprcflo)

status

show-repo-hook-status (srhs)

show-roots (sr)

show-writable-repo-override-config-filename (swrocf)

toggle-remote-debug (trd)

unittest

version

Figure 2 Sample Output: evnadmin

## Administration Commands

The most commonly used commands for day-to-day administration of Enversion are listed in this section.

### evnadmin create

Creates a new Subversion repository with Enversion enabled, and (optionally) prime the initial repository layout (using svnmucc).

### evnadmin analyze

Before Enversion can be enabled against an existing repository, the repository must be analyzed using this command. This will process every revision of the repository and construct the necessary roots in order for automatic root detection to be done once the hooks are enabled.

### evnadmin enable

Enables Subversion against an existing repository. This will analyse the repository first if necessary. Enabling Enversion means that the pre and post-commit hooks become active, such that invalid commits will be blocked at the pre-commit stage, and root-mutating commits will alter the evn:roots revision properties at the post-commit stage.

### evnadmin disable

Disable Enversion for the given repository. This alters the hooks such that Enversion is no longer invoked during pre and post-commit. This would typically be called only in extra-ordinary circumstances where Enversion is failing or raising unexpected exceptions and normal, valid commits aren’t being let through.

Once Enversion is enabled against a repository, it should not be disabled as long as commits can be made. Disabling Enversion will mean that root validation will not take place, nor will evn:roots revision properties be maintained until Enversion is re-enabled (and analysis is run against the revisions that were committed whilst it was disabled).

### evnadmin status

This is an alias for `evnadmin show-repo-hook-status`, and provides a means to quickly display whether or not Enversion has been enabled for a given repository.

## Customization Commands

This section describes the commands that you would run against an existing Enversion-enabled Subversion repository in order to customize the behaviour of major features.

### evnadmin set-repo-component-depth

This changes the component depth of a repository between single and multi (or disables it completely). See section *3 Repository Layout: Single vs Multiple Component* for more information on repository depth/layout.

### evnadmin get-repo-component-depth

Display the current component depth of a repository (single, multi or disabled).

### evnadmin set-repo-custom-hook-class

This command changes a repository’s custom hook class. It validates that the class can be loaded and is a subclass of evn.custom\_hook.CustomHook, then writes the custom-hook-class configuration file property to the writable configuration file for the repository. This is the recommended way to alter a repository’s custom hook class.

### evnadmin get-repo-custom-hook-class

Displays the current custom hook class for a given repository.

## Configuration Commands

These commands interact with the configuration files used by Enversion.

### evnadmin dump-config

### evnadmin dump-default-config

### evnadmin dump-modified-repo-config

### evnadmin dump-repo-config

### evnadmin show-possible-config-file-load-order

### evnadmin show-possible-repo-config-file-load-order

### evnadmin show-actual-config-file-load-order

### evnadmin show-actual-repo-config-file-load-order

### evnadmin show-writable-repo-override-config-filename

## Root Management Commands

### evnadmin show-roots

# Repository Layout: Single vs Multiple Component

Enversion has built-in support for two types of repository layouts: single-component layout and multiple-component layout. This is also referred to as component depth.

A single-component repository has a depth of 0 and would have a layout where the standard tags, branches and trunk directories are at the root of the repository:

/tags/

/v1.0/

/v1.1/

/branches/

/v1.x/

/trunk/

Figure 3 Example Single-Component Layout

A multi-component repository has a depth of 1 and would have a layout where the tags, branches and trunk directories live one level under the root directory – with the actual root directory being the name of the component:

/widget/

/tags/

/v1.0/

/v1.1/

/branches/

/v1.x/

/trunk/

/gadget/

/tags/

/v2.0/

/branches/

/v2.x/

/trunk/

Figure 4 Example Multiple-Component Layout

# Configuration Reference

## [main]

### blocked-file-extensions-iregex

|  |  |
| --- | --- |
| Data Type | String |
| Default Value | \.(com|ocx|mdb|dll|war|jar|so|exe|o|bin|iso|zip|tar|tgz|dat|tar\.gz|msi|msp|7z|pkg|rpm|nupkg|deb|dmg)$ |
| Since | 0.2.15 |
| Description | If set, a case-insensitive regular expression search is performed against each new path in a commit, and if a match is found, that path is flagged with the *BlockedFileExtension* error.  This functionality can be disabled with an empty string, e.g.:  blocked-file-extension-iregex = |
| Gotchas | 1. Care must be taken to ensure a valid regular expression is provided as the value is not validated when overridden. 2. Matches against any path name (i.e. files and directories). 3. The test is only done against newly added paths. Renaming an existing, valid path, to a path that matches the regex will not be blocked. (This was an oversight during initial implementation of the feature. It will likely change in future releases if feedback indicates that the behaviour should apply to renames as well.) |

Table 3 Configuration: blocked-file-extension-iregex

### custom-hook-classname

|  |  |
| --- | --- |
| Data Type | String |
| Default Value | evn.custom\_hook.DummyCustomHook |
| Since | 0.2.14 |
| Description | This is the name of a Python class that Enversion will create an instance of during pre and post-commit hooks. It is a literal string value that should represent a fully-qualified class name (i.e. include module prefix). The Python environment should be set up such that Enversion can import the containing module, as this is how the class name is resolved. (See [evn.util.load\_class](https://github.com/enversion/enversion/blob/master/lib/evn/util.py#L471) for the exact code used to load the class.)  The value of this property must always resolve to a valid class name that derives from [evn.custom\_hook.CustomHook](https://github.com/enversion/enversion/blob/master/lib/evn/custom_hook.py#L4), which is why the default value refers to a dummy hook class that doesn’t do anything in its pre\_commit and post\_commit callbacks.  As an invalid class name will prevent hooks from running, it is recommended that this property be set via the evnadmin set-repo-custom-hook-class command. |

Figure 5 Configuration: custom-hook-classname

# Errors, Warnings and Notes

## Errors

### BlockedFileExtension

|  |  |
| --- | --- |
| Format | “blocked file extension” |
| Controlled By | blocked-file-extensions-iregex |
| What It Means | The affected file matched the regex specified above. |

Table 4 Error: BlockedFileExtension