Repositories

Before reading this page, it is highly recommended you read repositories concept page.

What is a Repository?

A repository in Continua CI is a reference to an external Version Control System (VCS) that provides a way for Continua CI to interact with that VCS. Once a repository has linked a VCS to a configuration, all builds created by that configuration can then access the VCS and use it's files within the build process. Before a build is executed, Continua will check your VCS for any new changes, get the latest version and distribute the latest version to the Continua Agents that will run the build.

By using triggers, Continua can automatically trigger a build every time a change is committed to your VCS.

Additional Repository Information

- Repository Scope
- Repository User Mappings
- My Changes
- Repository Types

Repository Types

Continua CI uses multiple repository types when dealing with various VCSs. Each VCS that is supported by Continua has a matching repository type. For example, if you want to link a mercurial repository to Continua then you would have to create a mercurial repository. Each repository type requires different information to connect to their respective VCS. For more information on a specific type of repository, see the Repository Types section.

Repository Scope

Repositories can be created in Continua at either the Global, Project or Configuration level. Check out our Repository Scope page for more information on how to assign a repository to the various scope levels and why this distinction is a good idea.

Finding Repositories

Depending on the repository scope, repositories can be found in different areas of Continua CI:

Administration Repositories Section

The repositories administration page can be found in the **Administration section**, under **Continuous Integration (CI Server)**. On this page you can edit and delete every repository that exists in Continua CI, regardless of the repository scope.

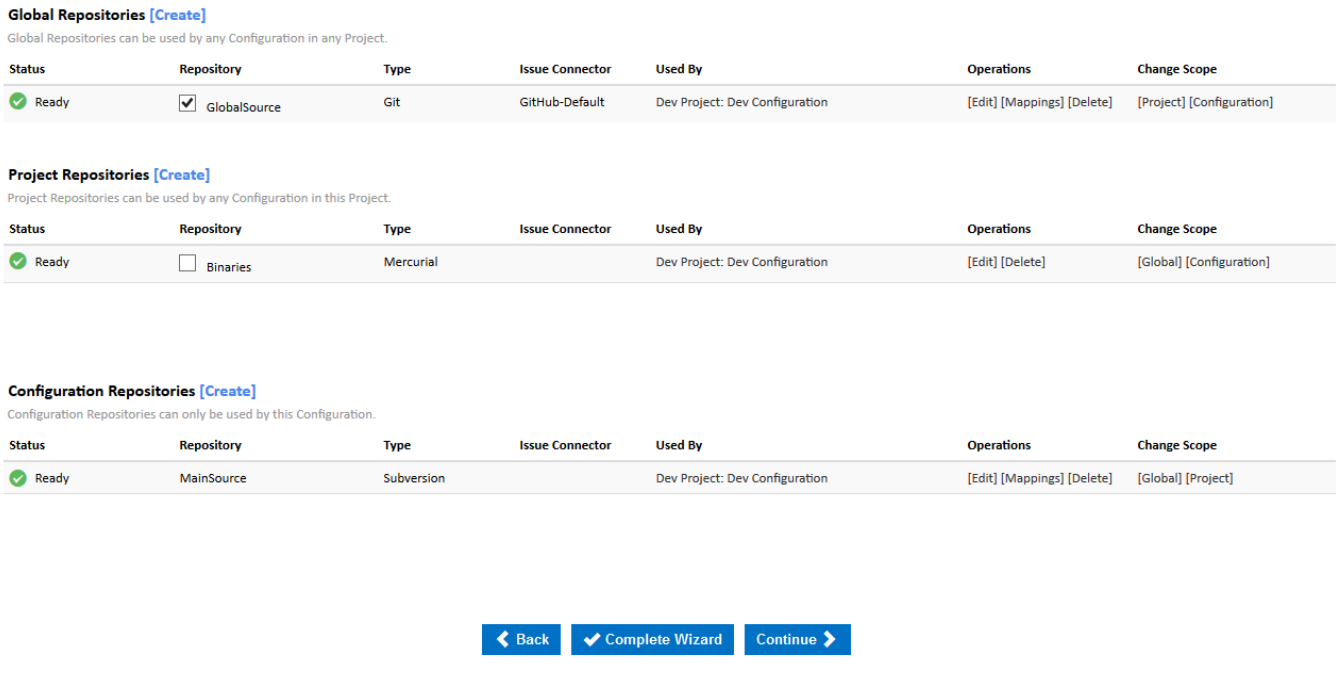
Note that any new repositories that are created in the Administration section will automatically be assigned to the Global scope, while existing repositories will keep their original scope.

The Administration repository page also allows you to reset repositories. More information on resetting repositories can be found in the **Resetting a Repository** section below.

Project Wizard Repositories Section

The Project Wizard repositories page can be found by editing a specific project and navigating to the repository section. This page allows you to create, edit and delete both project and global repositories.

Configuration Wizard Repositories Section

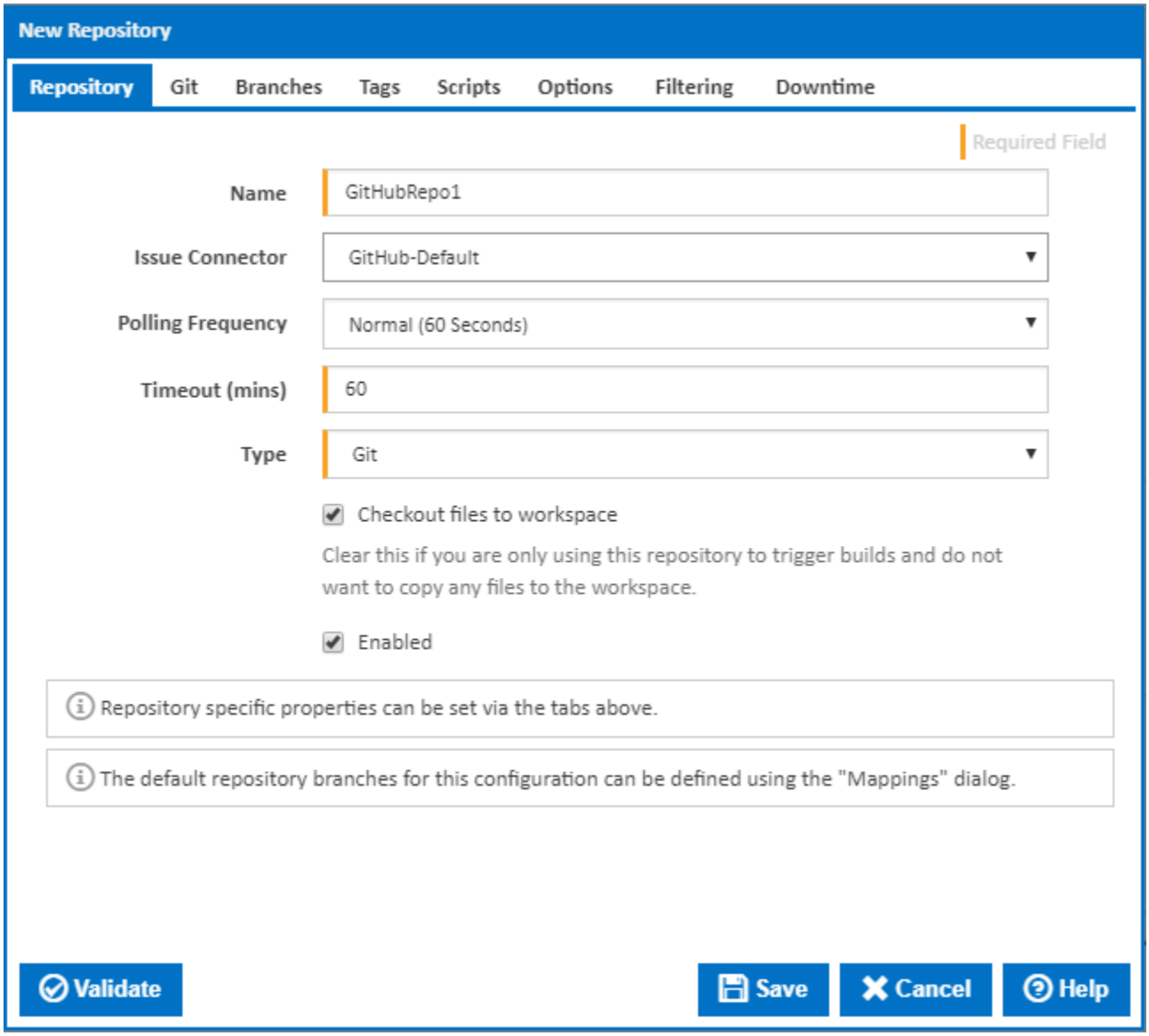


The Configuration Wizard repositories page can be found by editing a specific configuration and navigating to the repository section. This page allows you to create, edit and delete project, global and configuration repositories. You can also define default branches and branch mappings via the Mappings link for branch-aware repositories.

Through this page you can also assign project and global repositories to a specific configuration. Repositories can be assigned by ticking the Repository checkbox in either the Global Repositories table or Project Repositories table (In the figure above, **GlobalSource** has been assigned to this configuration while **Binaries** has not).

Creating and Editing Repositories

Repositories can be created and edited via the pages listed above.



Name

The name of this repository. Note that the repository name cannot contain spaces

Issue Connector

Issue Connectors monitor your commit messages and links any bug fixes to your issue tracking system. Selecting an issue connector on the repository means that Continua will monitor all commit messages from this repository for issue ids and then attempt to link these issues to your issue tracking system.

By default Continua comes with standard issue connectors that link to some of the more common issue tracking systems. Issue connectors can be created and edited under the administration section. For more information, view the Issue Connectors section.

Polling Frequency

Specify how often this repository should be checked for any new commits / check-ins.

The polling frequency can be set to **Never (Manual)** which means Continua will never check for changes. See the **Manually Polling for Changes** section below for more details on how to call manual polls.

Timeout (mins)

Specifies the timeout period, in minutes, for any processes where Continua interacts with this repository. Note that some processes can take a long time to execute, such as pulling an entire repository.

If a repository times out then Continua will put the repository into an errored state. See the **Errored Repositories** section below for more details on how to resolve an errored repository.

Type

Specifies the type of repository we are creating or editing, eg. Mercurial or Git. Once a repository type has been selected, additional repository type specific properties will be added to the dialog. These properties can be accessed through the tabs located at the top of the dialog.

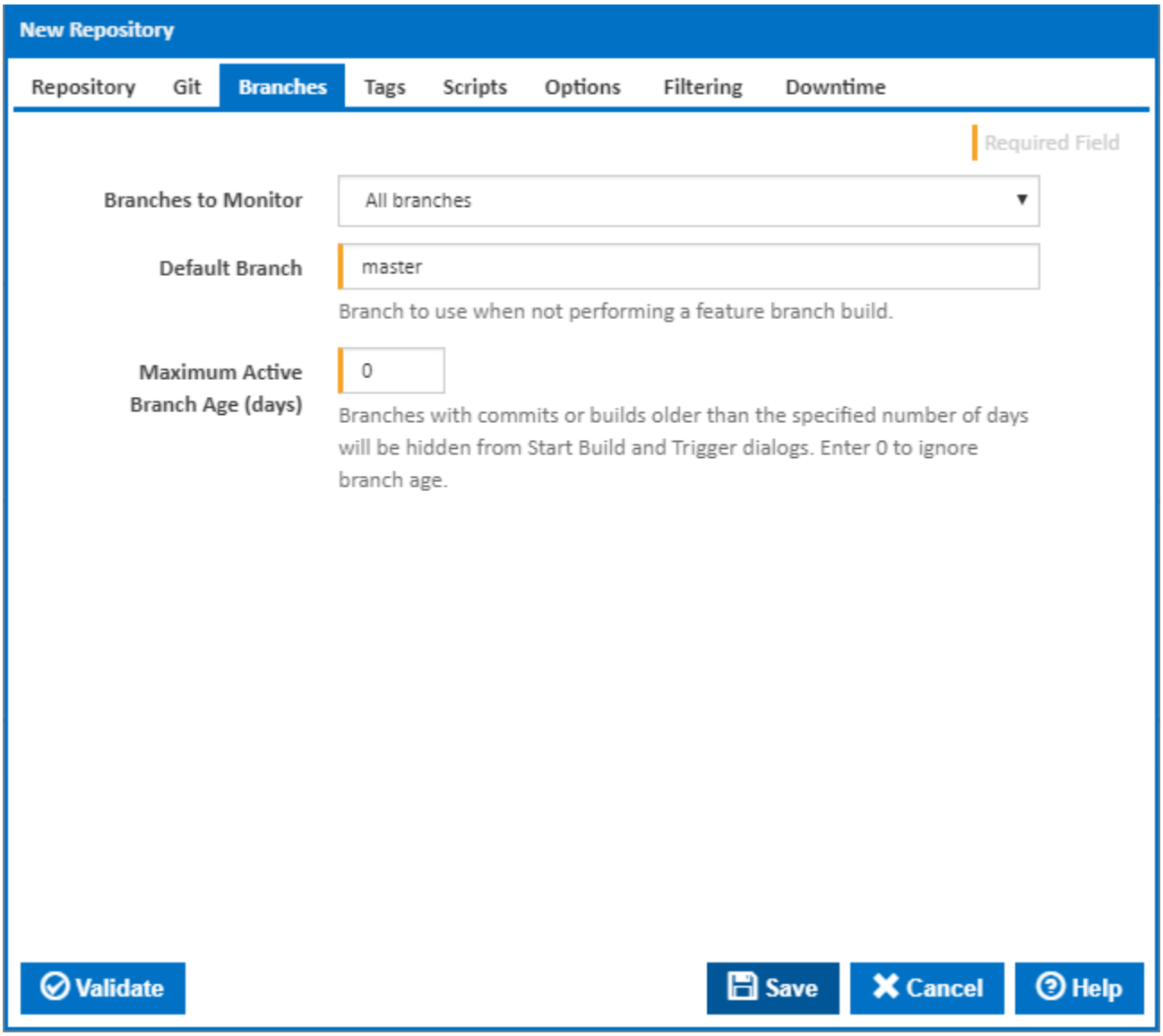
Checkout files to workspace

Clear this if you are only using this repository to trigger builds and do not want to copy any files to the workspace.

Enabled

Enable or disable a repository. View the **Disabled Repositories** section below for more information on disabled repositories.

Branches

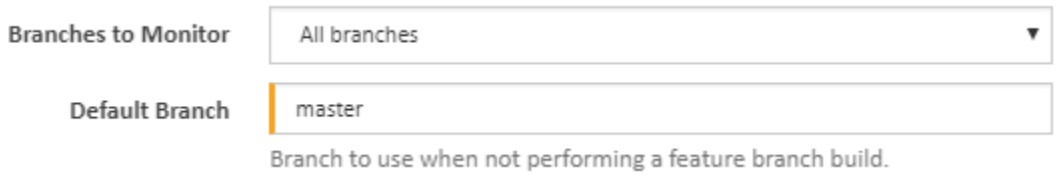


For branch-aware repositories the Branches tab allows you to refine how branches are handled by Continua. This setting affects which part of the given repository Continua looks for changes (useful in large repositories).

Options here include, a single branch, all branches, or branches with a specific pattern.

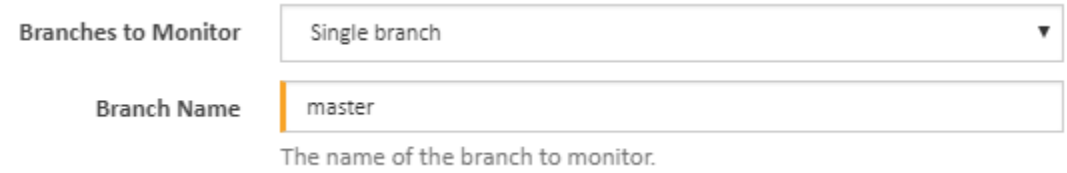
All Branches (the default for branch-aware repositories)

Consider all branches in the repository.



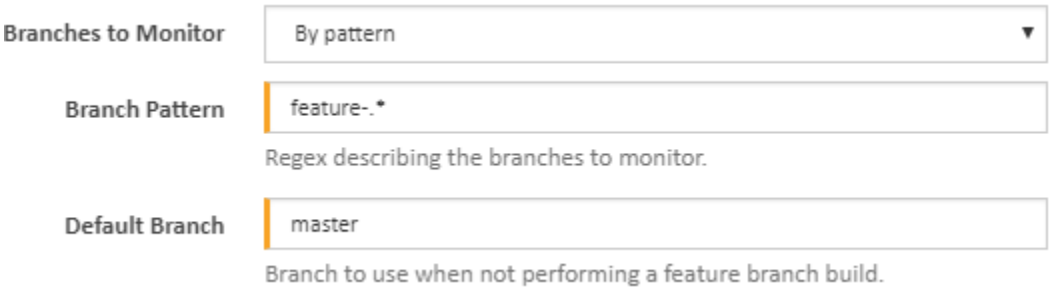
Single Branch

Consider the given branch only. The branch name must match exactly.



By Branch Pattern

Consider only branches which meet a specific pattern (using the supplied regex).

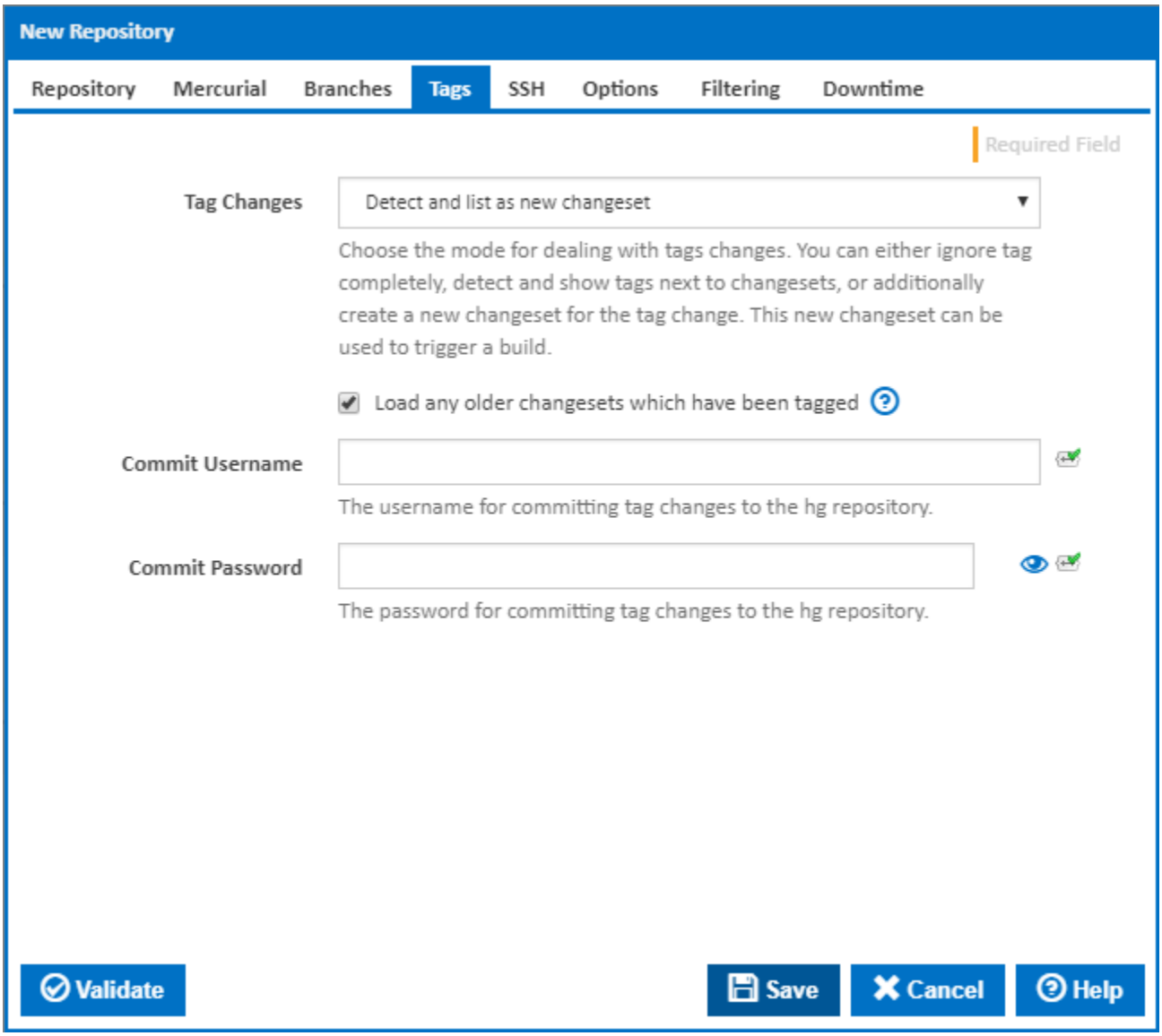


Maximum active branch age

You can also specify a maximum branch age. This allows you to exclude older branches from the selection list when starting a new build or defining a trigger.

Tags

For Tag-aware Repositories (Git, Mercurial and Subversion), the Tags tab sets how Continua handles changeset tags.



Tag changes

The options here include: 'Ignore', 'Detect', 'Detect and list as new Changeset'

Detect and list as new changeset (the Default for tag aware repositories)

Detect all tags for the given repository and treat them as a new changeset.

Ignore Tags

Do not consider Tags as changes.

Detect Branches

Detect tags and show tags next to changeset.

Commit / Push Username

The username to use when committing, or pushing for Git, tags to the repository (tags are committed via a Build Event Handler)

Commit / Push Password

The password to use when committing, or pushing for Git, tags to the repository (tags are committed via a Build Event Handler)

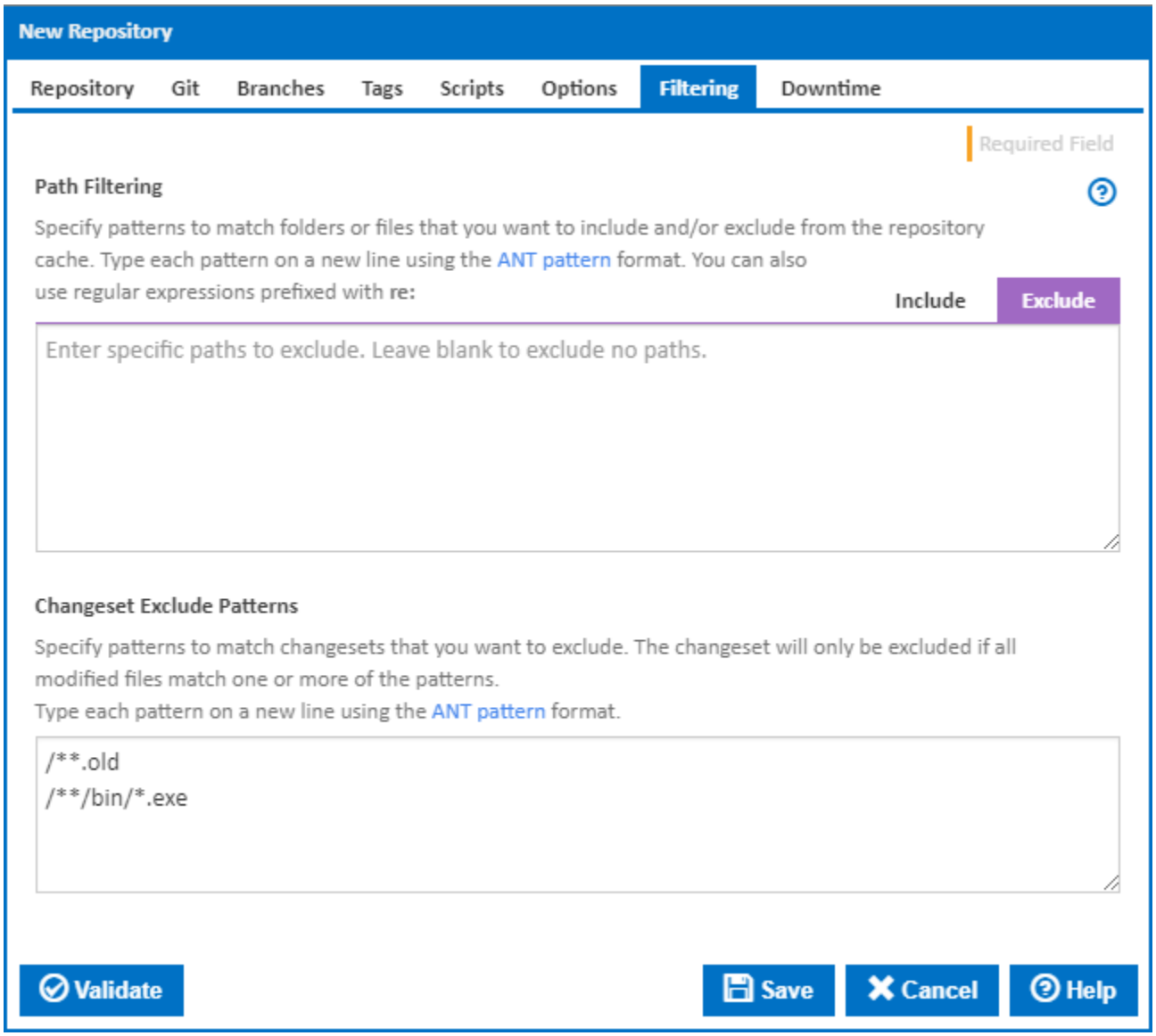
Push Repository URL (Git repositories)

The URL for pushing tags to the remote Git repository. If this is left blank the Repository URL will be used.

Tags Path (Subversion repositories)

The path the root directory for your tags

Filtering



Path Filtering

Specify patterns to match folders or files that you want to include and/or exclude from the repository cache. Type each pattern on a new line using the ANT pattern format. You can also use regular expressions prefixed with **re:**

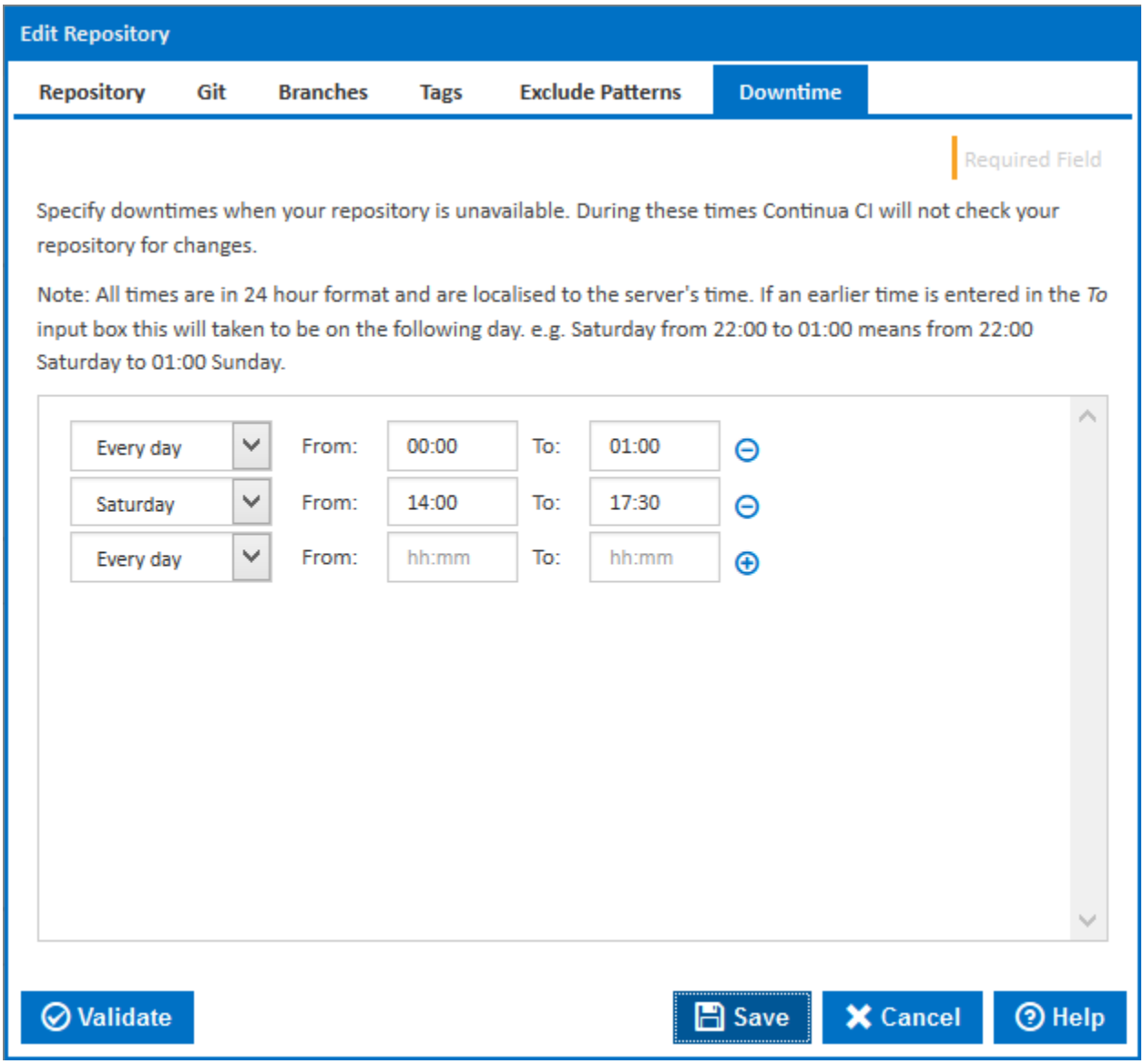
Changeset Exclude Patterns

Exclude patterns allow you to ignore incoming changesets for a repository based on the files that were changed. Changesets will be ignored if **every** updated file in the changeset matches your exclude patterns. For example, you could set your exclude patterns up so that if any old files (files that end in .old) were changed then Continua will ignore that changeset. Note that if another file was also modified in that changeset that **does not** match the exclude pattern, then the changeset will still be added to Continua.

Every exclude pattern must begin on a new line.

Exclude patterns use Ant Pattern formatting.

Downtime

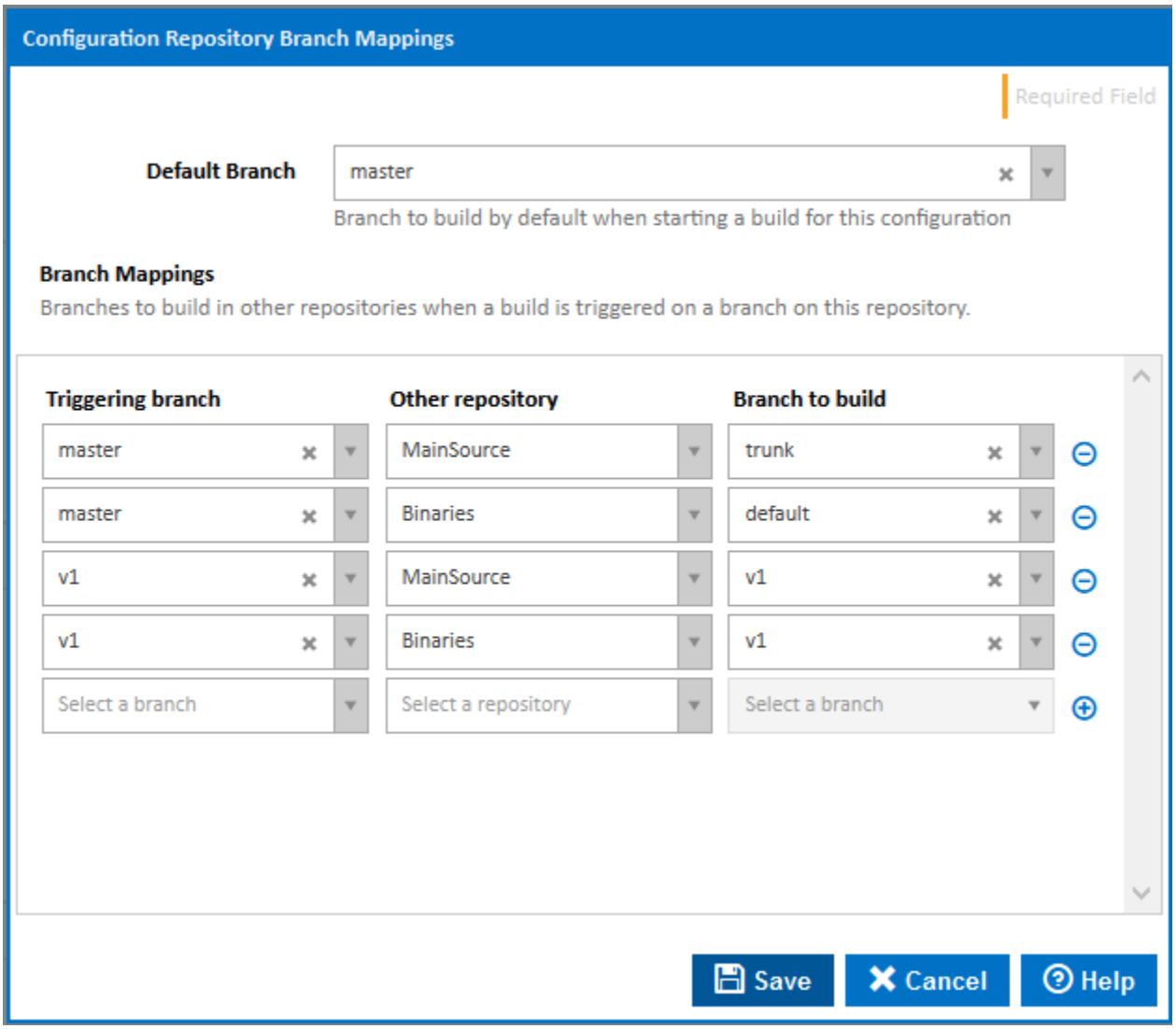


Setting downtime periods on your repository tells Continua CI that your repository should not be contacted during these periods. This allows you to take repositories offline without causing Continua to throw errors regarding contacting your repository.

Downtimes can be set to happen on a specific day of the week, on weekdays, weekends or every day.

All times are in 24 hour format and are localised to Continua's server time.

Configuration Repository Branch Mappings



Branch mappings can be accessed from the Configuration Wizard Repositories page by clicking on the Mapping link for each branch-aware repository.

Default Branch

This is the branch that is built for this repository when you run a build for the current configuration using the Quick Start Build button. It is also the default branch listed for this repository in the Queue Build dialog. This will override the default branch for the repository.

Triggers will also use the configuration default branch unless the branch is overridden in the trigger settings.

Branch Mappings

The branch mappings are mainly used to define which branches are built for a Repository Trigger. Branches are specified for each repository according to the triggering branch. These will override the default configuration branches for the repository.

Branch mappings can also be loaded into the queue build dialog clicking on a button next to the triggering repository.

Resetting a Repository

Resetting a repository forces Continua to delete the local repository cache for the specified repository and pull the latest changesets from your repository. When a repository is reset, Continua will get the latest changeset for each branch in that repository.

If your repository is in an errored state then resetting it may fix the issue.

Repositories can be reset through the **Administration Repositories page**.

Errored Repositories

A repository will be in an errored state if there are any issues when Continua attempts to get the latest changesets. If a repository is in an errored state then you can try to **resolve** the issue through the **Administration Repositories page**. Resolving a repository will reinitialise it and force to get the latest changesets. If you resolve a repository and it is still in an errored state, then you may need to reset the repository.

If resolving and resetting does not fix your repository then you should check your repository settings.

Disabled Repositories

By disabling a repository you are telling Continua to stop interacting with it. If a repository is disabled then Continua will not check for new changesets and any references to that repository will fail during the build process. For example, if you reference a disabled repository's path by using the following syntax **$Source.myRepo.Path$**, then at build time this will not be resolved, resulting in a failed build. As far as the build engine is concerned, the repository doesn't exist which means it cannot get the path value from the repository.

Manually Polling for Changes

Continua can be instructed to manually poll your repository by calling the **Manual Poll Url** that is displayed under the Polling Frequency property. Note that the Manual Poll Url property will only be displayed if **Polling Frequency** is set to **Never (Manual)**.