# The Trac Ticket Workflow System

The Trac ticket system provides a configurable workflow.

## The Default Ticket Workflow

When a new environment is created, a default workflow is configured in your trac.ini. This workflow is the basic workflow, as specified in ?basic-workflow.ini:

Enable JavaScript to display the workflow graph.

## Additional Ticket Workflows

There are example workflows provided in the Trac source tree, see ?contrib/workflow for `.ini` config sections. One of those may be a good match for what you want. They can be pasted into the `[ticket-workflow]` section of your `trac.ini` file. However, if you have existing tickets then there may be issues if those tickets have states that are not in the new workflow.

Here are some ?diagrams of the above examples.

## Basic Ticket Workflow Customization

**Note**: Ticket "statuses" or "states" are not separately defined. The states a ticket can be in are automatically generated by the transitions defined in a workflow. Therefore, creating a new ticket state simply requires defining a state transition in the workflow that starts or ends with that state.

In the `[ticket-workflow]` section of `trac.ini`, each entry is an action that may be taken on a ticket. For example, consider the `accept` action from `simple-workflow.ini`:

```

```

The first line in this example defines the `accept` action, along with the states the action is valid in (`new` and `accepted`), and the new state of the ticket when the action is taken (`accepted`).

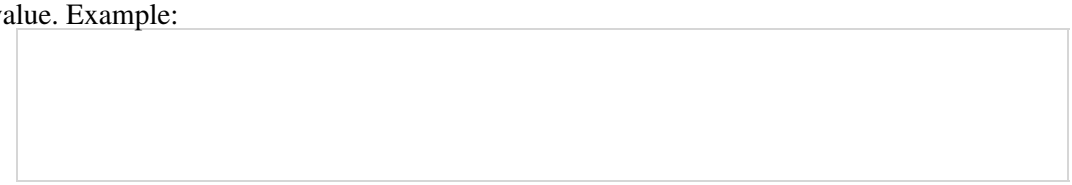
The `accept.permissions` line specifies the permissions the user must have to use this action. ?ExtraPermissionsProvider can define new permissions to be used here.

The `accept.operations` line specifies changes that will be made to the ticket in addition to the status change when the action is taken. In this case, when a user clicks on `accept`, the ticket owner field is updated to the logged in user. Multiple operations may be specified in a comma separated list.

The available operations are:

- **del_owner** -- Clears the owner field.
- **set_owner** -- Sets the owner to the selected or entered owner. Defaults to the current user. When `[ticket] restrict_owner = true`, the select will be populated with users that have `TICKET_MODIFY` permission and an authenticated session.

    - *actionname*`.set_owner` may optionally specify a comma delimited list of users that will be used to populate the select, or a single user. Groups and permissions may also be included in the list *(Since 1.1.3)*. When groups or permissions are specified the select is populated with all members of the group or all users that possess the permission.
- **set_owner_to_self** -- Sets the owner to the logged in user.
- **may_set_owner** -- Sets the owner to the selected or entered owner. Defaults to the existing owner. *(Since 1.1.2)*.
- **del_resolution** -- Clears the resolution field.
- **set_resolution** -- Sets the resolution to the selected value.
    - *actionname*`.set_resolution` may optionally be set to a comma delimited list or a single value. Example:



- **leave_status** -- Displays "leave as <current status>" and makes no change to the ticket.
- **reset_workflow** -- Resets the status of tickets that are in states no longer defined.

**Note:** Specifying conflicting operations, such as `set_owner` and `del_owner`, has unspecified results.

The example that follows demonstrates the `.label` attribute. The action here is `resolve_accepted`, but it will be presented to the user as `resolve`.



The `.label` attribute is new in Trac 1.1.3 and is functionally the same as the `.name` attribute, which is now deprecated. If neither `.label` or `.name` is specified, the action will be presented to the user as *resolve accepted*, the underscores having been replaced by whitespace (*Since 1.1.3*).

For actions that should be available in all states, `*` may be used in place of the state. The obvious example is the `leave` action:



This also shows the use of the `.default` attribute. This value is expected to be an integer, and the order in which the actions are displayed is determined by this value. The action with the highest `.default` value is listed first, and is selected by default. The rest of the actions are listed in order of decreasing `.default` values. If not specified for an action, `.default` is 0. The value may be negative.

There is one hard-coded constraint to the workflow: tickets are expected to have a `closed` state. The default reports/queries treat any state other than `closed` as an open state.

## Ticket Create Action

The ticket create actions are specified by a transition from the special `<none>` state. At least one create action must be available to the user in order for tickets to be created. The create actions defined in the default

workflow are:

### Ticket Reset Action

The special `_reset` action is added by default for tickets that are in states that are no longer defined. This allows tickets to be individually "repaired" after the workflow is changed, although it's recommended that the administrator perform the action by batch modifying the affected tickets. By default the `_reset` action is available to users with the `TICKET_ADMIN` permission and reset tickets are put in the *new* state. The default `_reset` action is equivalent to the following `[ticket-workflow]` action definition:

Since ?milestone:1.0.3 the `_reset` action can be customized by redefining the implicit action. For example, to allow anyone with `TICKET_MODIFY` to perform the `_reset` action, the workflow action would need to be defined:

## Workflow Visualization

Workflows can be visualized by rendering them on the wiki using the Workflow macro.

Workflows can also be visualized using the `contrib/workflow/workflow_parser.py` script. The script outputs `.dot` files that ?GraphViz understands. The script can be used as follows (your install path may be different):

```
$  /var/local/trac_devel/contrib/workflow/
$ ./showworkflow /srv/trac/PlannerSuite/conf/trac.ini
```

The script outputs `trac.pdf` in the same directory as the `trac.ini` file.

## Example: Adding optional Testing with Workflow

The following adds a `testing` action. When the ticket has status `new`, `accepted` or `needs_work`, you can choose to submit it for testing. When it's in the testing status the user gets the option to reject it and send it back to `needs_work`, or pass the testing and send it along to `closed`. If they accept it, then it is automatically

marked as `closed` and the resolution is set to `fixed`. Since all the old work flow remains, a ticket can skip this entire section.

## Example: Add simple optional generic review state

Sometimes Trac is used in situations where "testing" can mean different things to different people so you may want to create an optional workflow state that is between the default workflow's `assigned` and `closed` states, but does not impose implementation-specific details. The only new state you need to add for this is a `reviewing` state. A ticket may then be "submitted for review" from any state that it can be reassigned. If a review passes, you can re-use the `resolve` action to close the ticket, and if it fails you can re-use the `reassign` action to push it back into the normal workflow.

The new `reviewing` state along with its associated `review` action looks like this:

To integrate this with the default workflow, you also need to add the `reviewing` state to the `accept` and `resolve` actions:

Optionally, you can also add a new action that allows you to change the ticket's owner without moving the ticket out of the `reviewing` state. This enables you to reassign review work without pushing the ticket back to the `new` status:

The full `[ticket-workflow]` configuration will be:

## Advanced Ticket Workflow Customization

If the customizations above do not meet your needs, you can extend the workflow with plugins. Plugins can provide additional operations for the workflow, like code_review, or implement side-effects for an action, such as triggering a build, that may not be merely simple state changes. Look at ?sample-plugins/workflow for a few examples to get started.

But if even that is not enough, you can disable the ConfigurableTicketWorkflow component and create a plugin that completely replaces it. See also the ?AdvancedTicketWorkflowPlugin, which provides additional operations.

## Adding Workflow States to Milestone Progress Bars

If you add additional states to your workflow, you may want to customize your milestone progress bars as well. See the [milestone-groups] section.

## Ideas for next steps

Enhancement ideas for the workflow system should be filed as enhancement tickets against the ?ticket system component. You can also document ideas on the ?TracIdeas/TracWorkflow page.