

Notification Rules

Document ID: Q000060

Last Revised On: Tuesday, August 29, 2006

This article applies to the following:

Component(s):

Administrator

Solutions(s):

All

Summary

One of the fundamental features of the IssueNet platform is the ability to send notifications to contacts and resources when issues and tasks of interest have been created or changed. The change may be a result of prioritization, assignment, or from workflow rules. A notification may be sent to multiple parties using specific templates specified by the IssueNet solution.

Triggers and Transitions

IssueNet notifications are formatted and delivered using notification actions created in the IssueNet Administrator. There are two ways to execute notification actions to send notifications:

- A notification action can be executed based on a workflow transition.
- A notification can also be executed using an IssueNet trigger.

Notification actions are executed by workflows when a workflow transition is executed. Notifications based on changes to items, such as a change to an issue priority, can be executed anytime an item is created, updated, or deleted.

If you want to send a notification to a contact based on a specific workflow transition, you should add the notification action to the state or transition. For example, if you want to notify a contact based on the **Reject** transition in a **Review Issue** workflow to indicate that it has been rejected, you will want to click on that transition or state in that workflow and add that to the transition using the workflow property window.

If you do not want to send notifications not based on workflow transitions, you will want to use triggers to execute notification actions based on other events. A trigger is an item which can execute actions, such as sending notifications, when a user performs an action like inserting or updating an item. Each trigger is composed of three elements:

- The event that causes the trigger to execute. Triggers can be executed when items are inserted updated or deleted.
- Actions that the trigger will execute.
- A condition which determines if the trigger will execute the actions associated with it.

Triggers are used to execute a variety of actions in IssueNet. However, for the purposes of notification, you will almost always want to select the Before Insert or Before Update event. The reason is that the Before Insert and Before Update events allow you to easily compare values of an item before and after an insert or update, asynchronous events are only executed by the Monitor service, and the After Update and After Insert events do not allow you to compare values before and after update. The reason for using the Before Update and Before Insert events is that IssueNet keeps a copy of an item value for comparison to the new value before the update and that value is discarded after update.

Using Conditions in Trigger based Notifications

Conditions are used in a wide variety of contexts in IssueNet and provide an enormous degree of flexibility in the types of rules they can be used to implement. Accordingly, complete documentation of the uses and types of conditions is beyond the scope of this article. However, there are patterns that many conditions used in notification rules follow. Understanding these common types of conditions will allow you to create the majority of notification rules you will want to implement.

For trigger based notifications the condition is the most crucial element since it is primarily responsible for determining when a notification action is executed. In most environments there are two typical types of notifications:

- Sending a notification when an item is created with a particular property. For example, managers want to receive notifications when issues with a priority of **Urgent** are submitted.
- Sending a notification when a property of an existing item has changed – typically from one specific value to another. For example, sending a notification to the assigned resource to a task when the assignment changes.

In both of these notification examples we only want to send the notification when a particular type of item is created or updated – for example, an issue or task. If we create a rule that notifies based on priority value alone, it would notify based on both issues and tasks since both have a priority property. If you want to limit the notifications to a specific type of item, you need to add a statement to the condition that specifies the type of item, or class, the notification is to be sent about. To limit the notification to a specific item type we will start by adding a statement to a condition like the following:

If \$(CurrentObject.ClassID) is kind of "dae296e5-b651-48ec-ad0a-6da360167384"

In this example we used the condition editor to specify that we want to identify a class by ID and then used the Class option in the Type box of the Value section to allow us to pick a class and have the editor fill in the GUID identifier. We also used the “is kind of” operator in the Is box to specify that we wanted to use the object ID to check the type of class the item is. By doing this we limit the scope of the notification rule to a particular item type and its sub-types (classes). So, by picking issue as the class, this notification rule will apply to all issues and items that are members that are derived from issue. If we had picked the class “defect” the notification rule would be specific to members of that class.

Now that we have added a statement to the condition to limit the type of item the notification rule will apply to, we want to add additional statement(s) to specify the properties we are interested in. As described above, there are two typical types of property information rules apply to:

- When an item has a particular property value – For example, when an item is submitted with a priority of **Urgent**
- When an item has a property change – For example, when the assigned resource of a task changes.

In both instances you want to start by adding a new statement to the condition. You want to use the default operator that joins the statement which is "AND".

To use a condition to determine if an item has a property you will want to create a condition like the following:

If \$(CurrentObject.Priority) is equal to "Urgent"

This condition will determine if the current item has a priority of urgent. Other variables could be used to check different properties or properties of more specific items.

To use a condition to determine if a property has changed, you will want to use the **:Current** and **:Original** operators to compare values. To use a condition to determine if the assignment of a task changed you would want to add the following statement to your condition:

If \$(WorkflowTask.AssignID:Original) is not equal to \$(WorkflowTask.AssignID:Current)

This variable is created by adding the first conjunct using the **If** box, selecting **Variable** in the Value box and entering the second variable in the **Variable** box.

By taking advantage of the variable syntax for comparing current and original values, you can notify base on a change to any field – including text fields such as an item's description. You can also detect specific values changes. The following condition could be used to notify if an item's priority has changed from **Urgent** to any other value except **High**. This kind of condition would allow managers to be notified about unusual priority changes.

If \$(CurrentObject.Priority:Original) is equal to "Urgent" and \$(CurrentObject.Priority:Current) is not equal to "High"

Using Conditions for Workflow Based Notifications

Although conditions are most often used with trigger based notifications, you can also use conditions with workflow based notifications. The simplest way is to combine a notification action and a condition into a conditional action. For example, if you wanted to send a notification to a sales manager anytime a customer work order was transitioned to implementation, but only if the estimated cost recorded on the issue exceeded 5,000, you would combine a condition like the following into a conditional action that had the appropriate notification actions as one of its actions.

If \$(WorkflowIssue.EstimatedCost) is greater than or equal to "5000"

By combining this type of condition with a notification action into a conditional action, you can easily implement conditional notification rules per transition or state by adding the conditional action to that element.

Creating Notification Actions

Notification actions are created in the Administrator and each notification action consists of five basic elements:

- **Recipients** &ndash **Recipients** of a notification action can be specified as literal values or as variables. To specify a notification recipient using a literal e-mail address, simply click on the To.., Cc..,or Bcc.. buttons and use the Select Message Address dialog to enter an e-mail address or select a contact. The more common method is to specify a recipient using a variable. To specify a variable open the Select Message Address dialog and click on the Insert Variable button. There are many different types of variables you can use to send a notification &ndash so many that the full details are beyond the scope of this technote. However, common variables used to specify recipients would be:
 - **\$(Workflowtask.AssignID)** – Notifies the contact associated with the assigned resource for a task when transitioning a task.
 - **\$(CurrentObject.PrimaryContactID)** – Notifies the primary contact associated with an issue.
 - **\$(CurrentObject.SubmittedByID)** – Notifies the issue submitter.
 - **\$(WorkflowIssue.SubmittedByID)** – Notifies the submitter of an issue when transitioning a task the issue is linked to.
- **Object to Notify About** – In the Object box you specify which item you want the notification to be about. Because notifications are used as general rules you will always want to use a variable to identify the item you are sending the notification about. Like the use of variables to specify notification recipients, there are a wide variety of variables you can use to achieve different effects. However, there are some common variables that cover most types of notifications you will want to send. If you are using a trigger to send a notification, you would typically be sending a notification about the item you are updating or inserting. In that case you can always use the variable \$(CurrentObject). If you are, for example, transitioning a task, but would like to notify about the issue the task is linked to you could use the workflow variable \$(WorkflowIssue.ObjectID).
- **Template** – The **Template** is used to specify the notification template that will format the notification content. The template name is the path to the template within the notification template directory specified in the IssueNet Service Manager. For example, if your notification template directory contained a folder named **Task** that has a template named **NewTask**, you would identify that template in the notification rules as **Task\NewTask**.
- **Notification Queue** – The notification **Queue Name** is used to identify the IssueNet notification queue that will be used to store the notification until it is processed and sent to the mail server. Most organizations will have a single queue named **Default**.
- **Priority** – The **Priority** value allows you to specify the e-mail priority of the notification.

Additional Resources

[IssueNet Workflow Fundamentals](#)

[IssueNet Notification Configuration](#)
